EQUIVALENCES AND CALCULI FOR FORMAL VERIFICATION OF CRYPTOGRAPHIC PROTOCOLS

THÈSE N^{O} 4030 (2008)

PRÉSENTÉE LE 10 MARS 2008

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
Laboratoire de modèles et théorie de calculs
SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Johannes BORGSTRÖM

M.Sc. in Engineering, Royal Institute of Technology, Suède
et de nationalité suédoise

acceptée sur proposition du jury:

Prof. M. Odersky, président du jury
Prof. U. Nestmann, Prof. T. Henzinger, directeurs de thèse
Dr A. D. Gordon, rapporteur
Prof. V. Kuncak, rapporteur
Dr B. Victor, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2008

Summary

Security protocols are essential to the proper functioning of any distributed system running over an insecure network but often have flaws that can be exploited even without breaking the cryptography. *Formal cryptography*, the assumption that the cryptographic primitives are flawless, facilitates the construction of formal models and verification tools. Such models are often based on process calculi, small formal languages for modelling communicating systems.

The *spi calculus*, a process calculus for the modelling and formal verification of cryptographic protocols, is an extension of the pi calculus with cryptography. In the spi calculus, security properties can be formulated as equations on process terms, so no external formalism is needed. Moreover, the contextual nature of observational process equivalences take into account any attacker/environment that can be expressed in the calculus.

We set out to address the problem of automatic verification of observational equivalence in an extension of the spi calculus: A channel-passing calculus with a more general expression language.

As a first step, we study existing non-contextual proof techniques for a particular canonical contextual equivalence. In contrast to standard process calculi, reasoning on cryptographic processes must take into account the partial knowledge of the environment about transmitted messages. In the setting of the spi calculus, several notions of *environment-sensitive* bisimulation has been developed to treat this environment knowledge. We exhibit distinguishing examples between several of these notions, including ones previously believed to coincide. We then give a general framework for comparison of environment-sensitive relations, based on a comparison of the corresponding kinds of environment and notions of environment *consistency*. Within this framework we perform an exhaustive comparison of the different bisimulations, where every possible relation that is not proven is disproven.

For the second step, we consider the question of which expression languages are suitable. Extending the expression language to account for more sophisticated cryptographic primitives or other kinds of data terms quickly leads to decidability issues. Two important problems in this area are the knowledge problem and an indistinguishability problem called *static equivalence*. It is known that decidability of static equivalence implies decidability of knowledge in many cases; we exhibit an expression language where knowledge is decidable but static equivalence is not. We then define a class of *constructor-destructor* expression languages and prove that environment consistency over any such language directly corresponds to static equivalence in a particular extension thereof. We proceed to place some loose constraints on deterministic expression *evaluation*, and redefine the spi calculus in this more general setting.

Once we have chosen an expression language, we encounter a third problem,

which is inherent in the operational semantics of message-passing process calculi: The possibility to receive arbitrary messages gives rise to infinite branching on process input. To mitigate this problem, we define a *symbolic semantics*, where the substitution of received messages for input variables never takes place. Instead, input variables are only subject to logical constraints. We then use this symbolic semantics to define a *symbolic bisimulation* that is sound and complete with respect to its concrete counterpart, extending the possibilities for automated bisimulation checkers.

Keywords: Security Protocols, Formal Methods, Verification, Process Calculi, Operational Equivalence

Kurzfassung

Sicherheitsprotokolle sind eine Voraussetzung für die Funktionsfähigkeit jedweder verteilten Systeme, die über ein unsicheres Netzwerk laufen, weisen aber oft Lücken auf, die ausgenutzt werden können, sogar ohne die Kryptographie anzugreifen. *Formale Kryptographie*, die Annahme, dass kryptographische Primitive fehlerlos sind, erleichtert die Konstruktion von formalen Modellen und Verifizierungswerkzeugen. Solche Modelle basieren oft auf Prozesskalkülen, kleinen formalen Sprachen zur Modellierung von Kommunikationssystemen.

Das *Spi Kalkül*, ein Prozesskalkül zur Modellierung und formaler Verifizierung von kryptographischen Protokollen, stellt eine Erweiterung vom Pi Kalkül um Kryptographie dar. Im Pi Kalkül können Sicherheitseigenschaften als Gleichungen über Prozessterme formuliert werden, es ist also kein externer Formalismus nötig. Ausserdem berücksichtigt die kontextuelle Art von observationellen Prozessäquivalenzen alle Angreifer bzw. Umgebungen, die im Kalkül ausgedruckt werden können.

Wir beschäftigen uns mit dem Problem automatischer Verifizierung von Beobachtungsäquivalenz in einer Erweiterung des Spi Kalküls: Ein Kalkül für kanalübermittelnden Systemen mit einer allgemeineren Ausdruckssprache.

Als ersten Schritt untersuchen wir bestehende Beweistechniken, basierend auf Transitionssystemen, für eine bestimmte kanonische Kontextäquivalenz. Im Unterschied zu anderen Prozesskalkülen müssen wir bei Umgang mit die von Spi Prozessen generierten Transitionssystemen Wissen über die Umgebung mit einbeziehen. Um dieses Umgebungswissen zu erlangen, wurden einige Varianten von *umgebungsbewusster* Bisimularität für das Spi Kalkül entwickelt. Wir geben Beispiele, die die Unterschiede dieser Varianten zeigen, inklusive einiger bisher als gleich angenommener. Dann entwickeln wir einen allgemeinen Rahmen zum Vergleich von umgebungsbewussten Beziehungen, auf der Grundlage von einem Vergleich der entsprechenden Arten von Umgebungen und Begriffen von *Konsistenz*. Innerhalb dieses Rahmens stellen wir einen erschöpfenden Vergleich der verschiedenen Varianten von Bisimularität an, wo jede mögliche Beziehung, die nicht bewiesen wird, widerlegt wird.

Als zweiten Schritt betrachten wir die Frage, welche Ausdruckssprachen geeignet sind. Die Erweiterung der Ausdruckssprache zu anspruchsvollerer kryptographische Primitive oder andere Arten von Datentermen führt schnell zu Entscheidbarkeitsfragen. Zwei wichtige Probleme in diesem Gebiet sind das Wissensproblem und ein Nichtunterscheidbarkeitsproblem, die sogenannte *statische Äquivalenz*. Man weiss, dass aus der Entscheidbarkeit von statischer Äquivalenz die Entscheidbarkeit von Wissen in viellen Fällen folgt. Wir zeigen eine Ausdruckssprache, in der Wissen, nicht jedoch statische Äquivalenz, entscheidbar ist. Dann definieren wir eine Klasse von *Konstruktor-Destruktor* Ausdruckssprachen und beweisen, dass Umgebungskonsistenz über jegliche solche Sprache direkt der statischen Äquivalenz in einer bestimmten Erweiterung entspricht. Weiters legen wir einige lose Beschränkungen

auf die *Evaluierung* von determinischen Ausdrücken und redefinieren das Spi Kalkül in dieser allgemeineren Umgebung.

Sobald wir eine Ausdruckssprache gewählt haben, findet sich ein drittes Problem, das der operationalen Semantik von nachrichtenübermittelnden Kalkülen inherent ist: die Möglichkeit, jegliche Nachrichten zu empfangen, führt zu unendlicher Verzweigung bei Prozessinput. Um dieses Problem zu lindern, definieren wir eine *symbolische Semantik*, in der die Substitution von empfangenen Nachrichten durch Inputvariablen nie stattfinden. Stattdess unterliegen Inputvariablen nur logischen Beschränkungen. Wir nutzen diese symbolische Semantik dann dazu, um eine *symbolische Bisimularität* die "sound" und vollständig in Bezug auf ihr konkretes Gegenstück ist zu definieren, was weitere Möglichkeiten für automatische Bisimulationsprüfer eröffnet.

Keywords: Sicherheitsprotokolle, Formale Verifikation, Prozesskalküle, Operationeller Äquivalenz

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Security protocols are essential to the proper functioning of any distributed system running over an insecure network, such as the Internet. However, security protocols are notoriously hard to get right and often have flaws that can be exploited without breaking the cryptography [AN95] and with a low expenditure of resources.

In order to check if a protocol is vulnerable to this class of attacks, researchers turn to *formal cryptography* [DY83], working under the assumption that the cryptographic primitives are flawless. This allows the construction of formal models and tools for checking relevant properties of protocols [Low96]. Such models are often based on *process calculi* [Mil80], small formal languages for modelling communicating systems.

1.1 Background

The spi calculus, proposed by Abadi and Gordon [AG99] for the modelling and formal verification of cryptographic protocols, is an extension of the pi calculus [MPW92] with cryptographic operators and operations. The success of the spi calculus and its successors (notably the applied pi calculus [AF01]) is due to at least three reasons.

1. It is equipped with an operational semantics; thus any protocol described in the calculus may be regarded as executable.
2. Security properties can be formulated as equations on process terms, so no external formalism is needed to describe them.
3. The contextual nature of process equivalences avoids the need to explicitly model the attacker; they take into account any attacker that can be expressed in the calculus.

As seen in [AG99, DKR06], many correctness properties for cryptographic protocols are naturally expressed through equivalences between certain process terms.

To verify security properties expressed in this style, we need to choose a notion of equivalence. *Contextual* equivalences—two terms are related if they behave in the same way *in all contexts*—are attractive because the quantification over all contexts directly captures the intuition of an unknown attacker expressible within the spi calculus [AG98].

The most prominent notions of contextual equivalence are may-testing equivalence [dNH84] and barbed equivalence [MS92]. Their main distinction is linear time versus branching time: The former considers the possibility of passing a test after a sequence of computation steps; the latter has a more refined view, also comparing intermediate stages (cf. [Mil81]). Secrecy and authenticity are usually seen as trace-based properties and formulated in terms of testing equivalence; however, testing is not known to be sufficient for other security properties such as anonymity [CS02, DKR06].

Direct proofs of contextual equivalences are notoriously hard [AG99] due to the requirement of infinitary quantifications (usually quantifications over infinitely many process contexts). The standard solution to this problem is to work with an observational equivalence, based on the idea of an environment observing a pair of processes to see whether it may distinguish one from the other. The environment typically observes the labelled transition system derived from the operational semantics of processes. This approach is useful since in most process calculi, the two points of view of an observing environment and of an observed process are symmetric: any transition that a process can do according to its semantics is also observable by the environment. This symmetry is no longer valid in the case of the spi calculus.

In Figure 1.1, we show that the labelled transition system of the spi calculus, in contrast to the pi calculus, contains what one may call "meaningless transitions". The various labelled arrows represent the input $\xrightarrow{a\,b}$ of a name b along channel a, the output $\xrightarrow{\overline{a}\,b}$ of a message b along channel a, or the bound counterpart $\xrightarrow{(\nu b)\,\overline{a}\,b}$ for a fresh name b, and the (bound) output $\xrightarrow{(\nu bk)\,\overline{a}\,\mathrm{E}_k(b)}$ along channel a of a message $\mathrm{E}_k(b)$ representing the encryption of the clear-text b using the key k. The displayed transitions represent the possible observations about a process from the point of view of an environment interacting with the process. For example, an environment observing P might see the bound output $(\nu b)\,\overline{a}\,b$, upon which the environment performs a corresponding input operation. After the transition $P \xrightarrow{(\nu b)\,\overline{a}\,b} P'$, the fresh name b received by the environment may be used to interact with P' as seen in Figure 1.1. Essentially, once the environment receives a name, it may freely use it in interactions.

In the spi calculus there are also more complex transitions, such as in the transition from Q to Q' above, where the exchanged messages are encrypted. Note that both the key k and the datum b are bound, but when Q transmits the encrypted

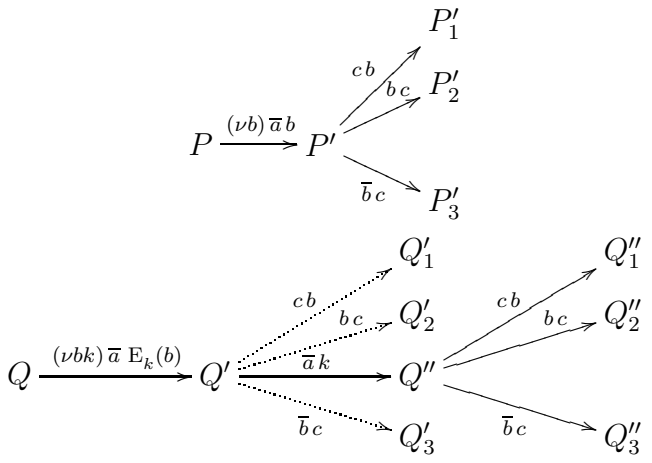

Figure 1.1: Environment Transitions in Pi and Spi Calculi

message neither the key nor the clear text b are accessible to the environment. Therefore, although present in the LTS of the process, none of the dotted transitions are observable by the environment of Q': since the environment does not know the key k, it cannot interact with Q' using the clear-text (the name b) hidden inside the cipher-text $\mathrm{E}_k(b)$, neither to communicate *on* the channel b nor to send back b to the process. However, these transitions become observable after the key k itself is sent to the environment, as in the transition from Q' to Q''.

This asymmetry between the possible and the observable actions of a process also makes the spi calculus highly interesting to study from a theoretical perspective.

1.1.1 Environment-sensitive Bisimulations

Labelled bisimilarity [Par81] is a coinductive observational equivalence that is often used as a proof technique for barbed equivalence. Its definition is based on the notion of simulation: A binary relation $\mathcal{R}$ on processes is a simulation if whenever $P\,\mathcal{R}\,Q$ and P has a transition $P \xrightarrow{\mu} P'$ then Q can *simulate* this transition by some $Q \xrightarrow{\mu} Q'$ such that $P'\,\mathcal{R}\,Q'$. Bisimilarity is then the greatest symmetrical simulation.

Unlike in the pi calculus, labelled bisimilarity is too strong a notion of equivalence for spi processes: It distinguishes between the (barbed equivalent) processes $(\nu k)\,\overline{a}\langle \mathrm{E}_k(M)\rangle$ and $(\nu k)\,\overline{a}\langle \mathrm{E}_k(N)\rangle$ whenever $M \neq N$, intuitively rendering encryption useless. This problem, and the observability issues related to Figure 1.1 as discussed above, were adressed by explicitly taking into account the *knowledge* of an environment about a process. As a means to capture the notion of environment knowledge, *environment-sensitive* bisimulations were developed for the spi calculus, in two main styles:

1. Abadi and Gordon [AG98] introduced *framed* bisimulation by imposing on every bisimulation pair a shared environment: A frame-theory pair representing the knowledge of the environment about the process pair. The *frame* is the set of names (channels, keys) that the environment has learned so far, while the *theory* is the set of pairs of non-name data items received from the pair of processes during the "bisimulation game". If all pairs of messages in the theory are indistinguishable, i.e., if environment has no means (e.g., decryption keys) to distinguish them, the environment is said to be *consistent*.
   Fenced bisimulation [EHHO99] is a refinement of framed bisimilarity, replacing one infinite existential quantification with an explicit algorithm.
2. Boreale et al. [BDP99] introduced another notion, under the generic name of environment-sensitive bisimulation, called "alley" bisimulation in the following. Here, each of the processes in a bisimulation pair is accompanied by an environment, which (roughly) lists the messages received from the process in the past. A pair of environments is *statically equivalent* if they validate the same logical formulae.

All of the above notions of bisimulation are, assuming that the observing environments know all free names of the related processes, sound approximations of barbed equivalence, and thus of may-testing equivalence.

In Chapter 2, after introducing the spi calculus, we formally highlight the differences and similarities between the above-mentioned notions of bisimulation by introducing a general framework for comparing environment-sensitive bisimilarities and an improved version of framed bisimulation, called *hedged* bisimulation.

1.1.2 Extending the Message Language

In the formal cryptography tradition, cryptographic messages are treated as terms modulo some equivalence relation, rather than as bit strings. The expression language used in Chapter 2 was chosen for minimality and only contains symmetric encryption with atomic keys. In order to model a cryptographic protocol we clearly need to extend this language, at the very least with tuples. However, for automated verification we need to be careful of how to choose the extension, in order to avoid that the properties we want to check (e.g., secrecy) become undecidable.

There are two main ways of specifying secrecy for a cryptographic protocol [CRZ07].

(1) One common approach is to see if the attacker can deduce the value of a secret parameter of the protocol, after some interaction with the protocol participants. This *disclosure*-based approach is taken in, e.g., [Low96, Sch96, KMM94].
(2) The other approach, used in the spi calculus, is to check whether the attacker can notice any difference between protocol runs with different values of the

secret parameter. This *indistinguishability*-based approach fits naturally into the process calculus framework of operational correspondence, is a standard notion of secrecy of cryptographic primitives [GM84], and is thus often used for protocol analysis in the probabilistic polynomial-time tradition [Mit01]. This approach can also be used to check other properties than secrecy, for instance by comparing an implementation of the protocol with an executable specification. At a given state in the protocol run, it gives raise to a static equivalence problem (cf. alley bisimulation above).

In Chapter 3, we show that in general, it is harder to decide static equivalence than disclosure. We then define a class of constructor-destructor message algebras where both problems are decidable and that permit a smooth generalization of the environment operations of Chapter 2.

1.1.3 Symbolic Semantics

Once we have chosen a non-contextual equivalence and a message language, we face an inherent problem with the operational semantics of message-passing process calculi: The possibility to receive arbitrary messages gives rise to an infinite number of "concrete" transitions.

Using a symbolic semantics, the substitution of received messages for input variables never takes place. Instead, an input prefix produces a single "symbolic" transition, where the input variable is only *indirectly* instantiated by collecting necessary constraints for later transitions.

In Chapter 4, we propose a symbolic structural operational semantics and a symbolic bisimulation for the full spi calculus. Symbolic bisimulation is significantly more complicated in the spi calculus than in value-passing CCS [HL95] or the pi-calculus [BD96, Lin94, Liu94]:

1. We must keep track of *when* an attacker has learned some piece of information so that it can only be used for instantiating inputs taking place *later on*;
2. The combination of scope extrusion and complex guards and expressions makes a precise correspondence to the concrete semantics challenging;
3. The cryptographic knowledge of the environment should be represented clearly and compactly;
4. Environment *inconsistency*, signaling that the environment has noticed a difference between the supposedly equivalent processes, must be carefully defined. A key issue here is how and when to evaluate expressions when defining the set of instantiating substitutions of a symbolic environment.

We choose hedged bisimulation as a starting point for our symbolic bisimulation since it offers a compact and clear knowledge representation. We prove the soundness

and completeness of the symbolic operational semantics for a large class of message languages, and the soundness of the symbolic bisimilarity with respect to hedged bisimulation.

1.2 Related Work

There exist several cryptographic process calculi other than the spi calculus. For generality we wanted to work with a channel-passing calculus. We chose to work with the spi calculus over the applied pi calculus due to the latter's lack of maturity at the time. Moreover, in the original definition of the applied pi calculus [AF01], channels could not be passed inside compound messages, limiting its usefulness.

Among the cryptographic process calculi without channel passing are Crypto-CSP [Low96, Sch96, SR01], which is an extension of CSP [AH85, Ros97], and Crypto-CCS [Mar99], which is an extension of CCS [Mil80]. Both Crypto-CSP and Crypto-CCS are used for property-based protocol-verification, where properties are defined as refinements of other CSP processes for Crypto-CSP and as logical predicates on states for Crypto-CCS.

1.2.1 Symbolic Techniques for Spi Calculi

Symbolic semantics have been exploited to implement bisimulation-checking algorithms for the pi calculus [San96, VM94]. Several symbolic semantics have also been defined for the limited setting of non-mobile spi calculi, where no channel-passing is allowed and channels often do not even exist. The first work along these lines was by Huima [Hui99]. Amadio and Lugiez [AL00] gave a symbolic model for reachability checking. Boreale [Bor01] defined a symbolic transition system and an algorithm for deciding whether a pair of events always occurs in the same order (cf. [WL93]). Boreale's tool (STA) was later extended [BB05] to other message languages, including asymmetric encryption both as a blackbox and using a more detailed model of modular exponentiation. Fiore and Abadi [FA01] defined another notion of symbolic traces, permitting complex keys (in contrast to the earlier work mentioned above where keys were restricted to be names). Martinelli [Mar02] gives a symbolic procedure for deciding intruder knowledge for a particular class of knowledge inference systems. The above-mentioned techniques are mainly intended for finite processes, and for that reason simply treat restriction (fresh name/key creation) as absence of initial environment knowledge, rather than as the actual generation of a previously unknown name at the appropriate point in the excecution of the process.

For the full spi calculus, where channel names and their communication pose challenges, the only earlier symbolic semantics that we are aware of was proposed by Durante et al. [DSV03]. The authors propose a tool (S^3A) based on an environment-

sensitive LTS, where the effects of structural and equality constraints are applied to the environment at each step. Like above, restriction is simply treated as uniqueness and absence of environment knowledge.

A more recent symbolic semantics and bisimulation than the original version of ours [BBN04] is due to Lü et al. [LCFW05], working in a non-mobile spi calculus without even internal communication. In that simpler setting they acheive the same soundness result as we had at the time; I am not aware of any complete version of their symbolic bisimilarity.

We prefer a pure process semantics that allows a more general expression language, has negation in the constraints, treats restriction as a syntactic binder, and lets us model the passing of channel names inside encrypted messages.

Recently, Delaune et al. [DKR07] have given a non-trivial extension of our symbolic bisimulation to the applied pi calculus, and proven it sound but incomplete with respect to barbed equivalence. Moreover, Johansson and Victor [JV07] have defined a sound and complete symbolic semantics and bisimulationfor the applied pi calculus, with an environment that is part of the process term and keeps track of knowledge, (dis)equality constraints and name freshness. We use a key idea from their completeness proof to define environment decompositions.

Other Spi Calculus Techniques By bounding the depth of messages, Hüttel gives a decision procedure [Hüt02] for framed bisimilarity in the case of finite processes. The time complexity of the procedure (doubly exponential, search tree branching factor $\gg 2^{2^{20}}$ for simple examples) makes it, as currently defined, of mainly theoretical interest.

Gordon, Jeffrey and Haack use typing to ensure authenticity [GJ04] and multi-level secrecy [GJ05] properties, also for timed protocols [GJ05, HJ06] and allowing unbounded processes.

Briais [Bri08] has specified hedged bisimilarity in Coq and given fully formal proofs for key results, allowing machine-checked bisimilarity proofs. A similar but unrelated work is by Kahsai and Miculan [KM07], that have specified both hedged and framed bisimilarity in Isabelle/HOL Nominal.

1.2.2 Calculi with Extended Message Algebras

The original spi calculus has symmetric encryption and pairing as constructors for composite messages, whereas the applied pi calculus permits the use of an arbitrary term algebra modulo an equivalence.

Cortier [Cor02, Cor03] extends alley bisimulation to a more general calculus than spi. The additions include both an extended expression language with public key encryption and more general guards. In this calculus, static equivalence is decidable

because of the regularity of the guards and an equational theory that only admits functions that are either one-way, completely invertible or partially invertible (i.e., encryption).

Blanchet [Bla01, AB05] gives a denotational semantics to an applied pi-like language by compiling it to Horn clauses. These can then be checked using logic programming techniques. This tool (ProVerif) has also been extended [BAF08] for certain kinds of equivalence-based secrecy checking. The possibility to define the message algebra to be used has permitted the use of ProVerif for verification of Internet protocols with complex message formats [BFGP03, LMBG05, BFGT06].

Some Decidability Results Studying the original spi calculus, Hüttel [Hüt02] proved that framed bisimilarity is decidable for finite spi, a spi calculus without complex keys, replication and recursion. He also showed that finite-control (i.e., recursion, parallel composition only at the top level) spi calculus is Turing-powerful, as opposed to finite-control pi calculus.

The concrete consistency problem has been solved for many different classes of message algebras and guard languages, including symmetric encryption [BDP02], public key encryption and hashing with general regular guards [Cor03], and more generally for a guard language only containing equality checks [AC06].

Turning to the symbolic case, Baudet [Bau07] gives an NP algorithm for the symbolic version of the knowledge problem in a more general setting than the spi calculus (called subterm-convergent rewrite theories). In that setting, Baudet also gives an NP algorithm for symbolic consistency, although for the case of guards without disjunction and negation.

1.3 Contributions

In Chapter 2, we study the different notions of bisimulations for the spi calculus that have been defined to date, and exhibit distinguishing notions between several of these notions, including ones previously believed to coincide. We then give a general framework for comparison of environment-sensitive relations, and interpret the above-mentioned counterexamples in this framework. We proceed to relate and compare the different bisimulations, based on a comparison of the corresponding kinds of environment, yielding a full account in the sense that every possible relation (from the above-mentioned framework) that is not proven is disproven. We finally extend this comparison to a category-theoretical (as opposed to a set-based) framework, using a corresponding interpretation of up-to techniques as quotient categories.

In Chapter 3, we give a full proof that for general message algebras, static equivalence is harder to decide than knowledge. We then define a class of constructor-destructor message languages where both of these problems are decidable. We show

how to extend the operations on hedges to this class of languages, and show that key properties of these operations still hold. To validate the definitions, we prove correspondence between consistency over any such message language, and static equivalence in an extension thereof. By the decidability of consistency, we immediately get that static equivalence and consistency are both decidable for this class of languages.

In Chapter 4, we give a general symbolic operational semantics for any spi calculus. We then use this semantics to define a symbolic bisimilarity, for spi calculi over constructor-destructor languages, that we prove to be both sound and complete with respect to its concrete counterpart.

1.4 The Spi calculus

The pi calculus [MPW92, Mil99, SW01] is a small language for modelling communicating and distributed systems, where communication channels can be generated and passed around. In its simplest form, processes P are built from the halted process $\mathbf{0}$, input prefixes $a(x)$ and output prefixes $\overline{a}\langle b\rangle$, parallel composition $P_1 \mid P_2$ and restriction (intuitively the generation of a fresh name) $(\nu a)\, P$. As an example, in $P := (\nu a)\,(a(x).\overline{b}\langle x\rangle.\,\mathbf{0} \mid (\nu c)\,\overline{a}\langle c\rangle.\,\mathbf{0})$, the process $(\nu c)\,\overline{a}\langle c\rangle.\,\mathbf{0}$ on the right-hand side of the parallel composition generates a fresh name c and transmits it on the hidden (restricted) channel a. The left-hand process $a(x).\overline{b}\langle x\rangle.\,\mathbf{0}$ receives a name on the channel a, later forwarding it on the public channel b. The process P can then perform the two transitions

$$(\nu a)\,(a(x).\overline{b}\langle x\rangle.\,\mathbf{0} \mid (\nu c)\,\overline{a}\langle c\rangle.\,\mathbf{0}) \xrightarrow{\tau} (\nu a)\,(\nu c)\,(\overline{b}\langle c\rangle.\,\mathbf{0} \mid \mathbf{0}) \xrightarrow{(\nu c)\,\overline{b}\,c} (\nu a)\,(\mathbf{0} \mid \mathbf{0}).$$

Here the label τ on the first arrow denotes internal communication, while the label $(\nu c)\,\overline{b}\,c$ denotes the transmission of a fresh name c on channel b.

In contrast to the pi calculus, the spi calculus offers next to mere names another kind of transmissible messages, namely *ciphertexts*, which are provided by the addition of primitive constructs to encrypt ($\mathrm{E}_k(M)$) and decrypt ($\mathrm{D}_k(M)$) data using shared-key cryptography.

The original spi calculus [AG99] also included tuples. Here we use this simple expression language since it is already sufficient to exhibit the differences between bisimilarities; we treat extensions of the language in Section 3.4.

We build on the same assumptions on the underlying system of shared-key cryptography as [BDP02], which read as follows:

1. *Perfect Encryption*: A ciphertext $\mathrm{E}_k(M)$, i.e., a message M encrypted under a key k, can only be decrypted using k. The only way to produce the ciphertext $\mathrm{E}_k(M)$ is to encrypt M under k. If k is secret, no attacker can guess or

forge k. An attacker possessing two different encrypted messages cannot tell whether they are encrypted with the same key, contain the same or related cleartexts or are completely unrelated.
2. There is enough redundancy in the structure of messages to tell whether decryption of a message with a given key has actually succeeded or not.
3. There is enough redundancy in the structure of messages to tell their role (name or compound ciphertext).
4. Compound messages are not admissible as encryption keys.

Assumption 3 is necessary since we only permit communications on channels corresponding to a name (i.e., a compound message cannot be used as a channel, in contrast to [DSV03, Tiu07]). To explicitly check for this distinction we have a guard $[\,F:\mathcal{N}\,]$, the semantics of which can be found in Table 1.4 on page 13.

1.4.1 Syntax

We assume a countably infinite set $\mathcal{N}$ of names. Names are untyped, meaning that the same name can be used as a channel, a key or the clear-text of a message. The lower case letters a, b, c, k, l, m, n are used to range over names. We use x, y, z to range over the infinite set $\mathcal{V}$ of variables. We let u, v, w range over $\mathcal{N} \cup \mathcal{V}$. While expressions F are formed arbitrarily using both constructors and destructors, messages M represent nestings of encryptions.

Logical formulae ϕ generalize the usual matching operator of the pi calculus by conjunction and negation. Moreover, the predicate $[\,F:\mathcal{N}\,]$ tests for the format of F, i.e., whether it evaluates to a plain name or not.

The syntax of messages, expressions, guards and processes is given in Table 1.1. Our syntax is similar to that of the applied pi calculus, without a *let* construct. This is beneficial since the set of messages $\mathcal{M}$ may not be closed under injective (α-)renaming in extensions of the message language (see Example 3.4.5).

When writing down terms, we use the convention that $(\nu b)\,,\ \phi$ and prefixing bind stronger than $+$ and that $+$ binds stronger than $|$, and let $+$ and $|$ be left associative. We generally omit $\mathbf{0}$ when after a prefix.

The names $\mathrm{n}(\cdot)$ resp. variables $\mathrm{v}(\cdot)$ of a term are the names resp. variables occuring in the term. Free and bound names and variables of process terms are inductively defined as expected (Table 1.2): the name a is bound in "$(\nu a)\,P$" and the variable x is bound in "$F(x).P$" and "$!\,F(x).P$". Two processes are *α-equivalent* if they can be made equal by conflict-free renaming of bound names and variables. We identify α-equivalent processes, except as noted below.

Substitutions σ are of two kinds: *instantiating* or *renaming*. Instantiating substitutions are idempotent functions $\left\{{}^{F}/_{x}\right\}$ from variables x to expressions F. Renamings are injective functions $\mathcal{N} \to \mathcal{N}$. Substitutions are applied to processes,

F, G	$::=$	u	$\mid \mathrm{E}_G(F)$	$\mid \mathrm{D}_G(F)$		expressions $\mathcal{E}$
M, N	$::=$	a	$\mid \mathrm{E}_a(M)$			messages $\mathcal{M}$
ϕ, ψ	$::=$	tt	$\mid \phi \wedge \phi$	$\mid \neg\phi$		guards $\mathcal{G}$
	$\mid$	$[\, G = F \,]$				(equality)
	$\mid$	$[\, F : \mathcal{N} \,]$				(is a name)
P, Q	$::=$	$\mathbf{0}$				processes $\mathcal{P}$
	$\mid$	$F(x).P$				(input prefix)
	$\mid$	$\overline{F}\langle F \rangle.P$				(output prefix)
	$\mid$	$!\, F(x).P$				(replicated input)
	$\mid$	$P + P$				(choice)
	$\mid$	$P \mid P$				(parallel)
	$\mid$	$(\nu a)\, P$				(restriction)
	$\mid$	ϕP				(boolean guard)

Table 1.1: Syntax of the Spi Calculus

expressions and guards in the straightforward way, obeying the usual assumption that capture of bound names is avoided through implicit α-conversion: for example, $P\{^F/_x\}$ replaces all free occurrences of x in P by F, renaming bound names and variables in P where needed. We write $\sigma\{^M/_x\}$ for $\sigma \cup \{(x, M)\}$, where $x \notin \mathrm{dom}(\sigma)$. To add several messages, we write $\sigma\{^{M_1}/_{x_1}, \ldots, ^{M_n}/_{x_n}\}$ for $\sigma \cup \{\, (x_i, M_i) \mid i = 1, 2, \ldots, n \,\}$ where the x_i are assumed to be pairwise different and not in $\mathrm{dom}(\sigma)$. The set of free names of a substitution is defined as $\mathrm{fn}(\sigma) := \mathrm{n}(\mathrm{range}(\sigma))$. If σ is a renaming, we define its support as $\mathrm{spt}(\sigma) := \{a \in \mathcal{N} \mid \sigma(a) \neq a\}$.

When $s_1, \ldots, s_k$ are terms (where k may be 0), we write "$\widetilde{s}$" as a shorthand for "$s_1, \ldots, s_k$". We write "$(\nu a, \widetilde{c})$" for "$(\nu a)\,(\nu \widetilde{c})$" and "$(\nu)$" for "" (nothing). When f is a function from terms to terms we write "$f(\widetilde{s})$" for "$f(s_1), \ldots, f(s_k)$". When f is a function from terms to sets of terms we write "$f(a, \widetilde{s})$" for "$f(a) \cup f(\widetilde{s})$" and "$f()$" for "$\emptyset$", e.g., $\mathrm{fn}(P, Q)$ stands for $\mathrm{fn}(P) \cup \mathrm{fn}(Q)$.

1.4.2 Semantics

The structural operational semantics for our spi calculus mostly resembles the one for the pi calculus, with three major differences:

- Since input and output prefixes contain arbitrary expressions, we must make sure that these expressions evaluate to a concrete message or channel name before performing the transition.
- Since a message may contain several different fresh names, the side conditions guaranteeing name freshness are more complex.

P	$\mathrm{fn}(P)$	$\mathrm{bn}(P)$
$\mathbf{0}$	$\emptyset$	$\emptyset$
$F(x).Q$	$\mathrm{n}(F) \cup \mathrm{fn}(Q)$	$\mathrm{bn}(Q)$
$\overline{F}\langle G\rangle.Q$	$\mathrm{n}(F) \cup \mathrm{n}(G) \cup \mathrm{fn}(Q)$	$\mathrm{bn}(Q)$
$!\,F(x).Q$	$\mathrm{n}(F) \cup \mathrm{fn}(Q)$	$\mathrm{bn}(Q)$
$Q_1 + Q_2$	$\mathrm{fn}(Q_1) \cup \mathrm{fn}(Q_2)$	$\mathrm{bn}(Q_1) \cup \mathrm{bn}(Q_2)$
$Q_1 \mid Q_2$	$\mathrm{fn}(Q_1) \cup \mathrm{fn}(Q_2)$	$\mathrm{bn}(Q_1) \cup \mathrm{bn}(Q_2)$
$(\nu a)\,Q$	$\mathrm{fn}(Q) \setminus \{a\}$	$\{a\} \cup \mathrm{bn}(Q)$
ϕQ	$\mathrm{n}(\phi) \cup \mathrm{fn}(Q)$	$\mathrm{bn}(Q)$

P	$\mathrm{fv}(P)$	$\mathrm{bv}(P)$
$\mathbf{0}$	$\emptyset$	$\emptyset$
$F(x).Q$	$\mathrm{v}(F) \cup (\mathrm{fv}(Q) \setminus \{x\})$	$\{x\} \cup \mathrm{bv}(Q)$
$\overline{F}\langle G\rangle.Q$	$\mathrm{v}(F) \cup \mathrm{v}(G) \cup \mathrm{fv}(Q)$	$\mathrm{bv}(Q)$
$!\,F(x).Q$	$\mathrm{v}(F) \cup (\mathrm{fv}(Q) \setminus \{x\})$	$\{x\} \cup \mathrm{bv}(Q)$
$Q_1 + Q_2$	$\mathrm{fv}(Q_1) \cup \mathrm{fv}(Q_2)$	$\mathrm{bv}(Q_1) \cup \mathrm{bv}(Q_2)$
$Q_1 \mid Q_2$	$\mathrm{fv}(Q_1) \cup \mathrm{fv}(Q_2)$	$\mathrm{bv}(Q_1) \cup \mathrm{bv}(Q_2)$
$(\nu a)\,Q$	$\mathrm{fv}(Q)$	$\mathrm{bv}(Q)$
ϕQ	$\mathrm{v}(\phi) \cup \mathrm{fv}(Q)$	$\mathrm{bv}(Q)$

Table 1.2: Free and Bound Names and Variables

- Guards generalize the standard matching construct with conjunction, negation and the possibility to check whether an expression evaluates to a channel name.

The set of (late input) actions $\mu \in \mathcal{A}$ is defined as $\mu ::= a(x) \mid (\nu\tilde{c})\,\overline{a}\,M \mid \tau$, where the bound names $\tilde{c}$ must be pair-wise different. We let $\mathrm{bv}(a(x)) := \{x\}$ and $\mathrm{bn}((\nu\tilde{c})\,\overline{a}\,M) := \{\tilde{c}\}$. For evaluation of expressions we have a partial function $\mathbf{e}(\cdot) : \mathcal{E} \rightharpoonup \mathcal{M}$ that is defined recursively as in Table 1.3. For guards we have a predicate $[\![\cdot]\!]$ that is defined in Table 1.4.

Operational Semantics

In Table 1.5, we give the transition rules for the late input semantics for closed processes ($\mathrm{fv}(P) = \emptyset$). We assume that no α-renaming of processes occurs during the derivation of a transition. Instead, we define α-equivalence on output transitions. If $P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P'$ and $\sigma : \{\tilde{b}\} \to \mathcal{N}$ is injective such that $(\mathrm{range}(\sigma) \setminus \mathrm{dom}(\sigma)) \cap (\mathrm{n}(a, M) \cup \mathrm{fn}(P')) = \emptyset$ then $(P \xrightarrow{\mu} P') =_\alpha (P \xrightarrow{(\nu\tilde{b}\sigma)\,\overline{a}\,M\sigma} P'\sigma)$.

$$G \overset{\mathbf{e}}{\mapsto} \begin{cases} a & \text{if } G = a \\ \mathrm{E}_k(M) & \text{if } G = \mathrm{E}_{F_2}(F_1) \text{ and } \mathbf{e}(F_1) = M \in \mathcal{M} \quad \text{and } \mathbf{e}(F_2) = k \in \mathcal{N} \\ M & \text{if } G = \mathrm{D}_{F_2}(F_1) \text{ and } \mathbf{e}(F_1) = \mathrm{E}_k(M) \in \mathcal{M} \text{ and } \mathbf{e}(F_2) = k \in \mathcal{N} \\ \bot & \text{if otherwise} \end{cases}$$

Table 1.3: Concrete Evaluation (Decryption)

$[\![tt]\!]$	is true	
$[\![\phi \wedge \psi]\!]$	is true	iff $[\![\phi]\!]$ and $[\![\psi]\!]$ are true
$[\![\neg\psi]\!]$	is true	iff $[\![\psi]\!]$ is not true.
$[\![F = G]\!]$	is true	iff $\mathbf{e}(F) = \mathbf{e}(G) \neq \bot$
$[\![F : \mathcal{N}]\!]$	is true	iff $\mathbf{e}(F) \in \mathcal{N}$

Table 1.4: Guard Satisfaction

Example 1.4.1 *We give a process Q having the transitions of Figure 1.1.*

$$Q := (\nu b, k)\, \overline{a}\langle \mathrm{E}_k(b) \rangle.(\overline{a}\langle k \rangle \mid \overline{b}\langle c \rangle + b(x) + c(x))$$

Definition 1.4.2 *A process P exhibits the (strong)* barb *b, written $P \downarrow_b$, if $\exists x, P'.\ P \xrightarrow{b(x)} P'$. We let $\Rightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$. The process P exhibits the* weak barb *b, written $P \Downarrow_b$, if $\exists P'.\ P \Rightarrow P'$ and $P' \downarrow_b$.*

Our target contextual equivalence is (strong) barbed equivalence. Since we are in a synchronous setting, it is sufficient to study input barbs. Hüttel [Hüt02] has showed that two-counter machines can be implemented in the spi calculus; this implies that even the existence of a weak barb is undecidable. For this reason, strong equivalences are more amenable to automated verification.

Definition 1.4.3 *A binary relation $\mathcal{R} \subset \mathcal{P} \times \mathcal{P}$ is a (strong) barbed bisimulation if whenever $P\ \mathcal{R}\ Q$,*

1. *If $P \xrightarrow{\tau} P'$ then $Q \xrightarrow{\tau} Q'$ such that $P'\ \mathcal{R}\ Q'$; and*
2. *If $Q \xrightarrow{\tau} Q'$ then $P \xrightarrow{\tau} P'$ such that $P'\ \mathcal{R}\ Q'$; and*
3. *For all $b \in \mathcal{N}$, if $P \downarrow_b$ then $Q \downarrow_b$; and*
4. *For all $b \in \mathcal{N}$, if $Q \downarrow_b$ then $P \downarrow_b$.*

We let barbed bisimilarity, $\dot{\simeq}$, be the union of all barbed bisimulations. Two processes P and Q are (strongly) barbed equivalent, written $P \simeq Q$, if for all processes R, $(P \mid R) \dot{\simeq} (Q \mid R)$.

$$(\textsc{out})\ \frac{\mathbf{e}(G) = a \quad \mathbf{e}(F) = M}{\overline{G}\langle F\rangle.P \xrightarrow{\overline{a}\,M} P} \qquad (\textsc{inp})\ \frac{\mathbf{e}(G) = a}{G(x).P \xrightarrow{a(x)} P}$$

$$(\textsc{com-l})\ \frac{P \xrightarrow{a(x)} P' \quad Q \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} Q'}{P \mid Q \xrightarrow{\tau} (\nu\tilde{b})\ \left(P'\{^{M}/_{x}\} \mid Q'\right)}\ \text{if}\ \{\tilde{b}\} \cap \mathrm{fn}(P) = \emptyset$$

$$(\textsc{com-r})\ \frac{P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P' \quad Q \xrightarrow{a(x)} Q'}{P \mid Q \xrightarrow{\tau} (\nu\tilde{b})\ \left(P' \mid Q'\{^{M}/_{x}\}\right)}\ \text{if}\ \{\tilde{b}\} \cap \mathrm{fn}(Q) = \emptyset$$

$$(\textsc{guard})\ \frac{P \xrightarrow{\mu} P'}{\phi P \xrightarrow{\mu} P'}\ \text{if}\ [\![\phi]\!] \qquad (\textsc{rep})\ \frac{\mathbf{e}(G) = a}{!\,G(x).P \xrightarrow{a(x)} P \mid !\,G(x).P}$$

$$(\textsc{par-l})\ \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\ \text{if}\ \ \mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset \qquad (\textsc{sum-l})\ \frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'}$$

$$(\textsc{par-r})\ \frac{Q \xrightarrow{\mu} Q'}{P \mid Q \xrightarrow{\mu} P \mid Q'}\ \text{if}\ \ \mathrm{bn}(\mu) \cap \mathrm{fn}(P) = \emptyset \qquad (\textsc{sum-r})\ \frac{Q \xrightarrow{\mu} Q'}{P + Q \xrightarrow{\mu} Q'}$$

$$(\textsc{open})\ \frac{P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P'}{(\nu c)\,P \xrightarrow{(\nu c\tilde{b})\,\overline{a}\,M} P'}\ \text{if}\ \ \mathrm{n}(M) \ni c \notin \{a, \tilde{b}\}$$

$$(\textsc{res})\ \frac{P \xrightarrow{\mu} P'}{(\nu c)\,P \xrightarrow{\mu} (\nu c)\,P'}\ \text{if}\ c \notin \mathrm{n}(\mu)$$

$$(\textsc{alp})\ \frac{P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P'}{P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,N} Q}\ \text{if}\ (P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P') =_{\alpha} (P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,N} Q)$$

Table 1.5: Operational Semantics

The definitions of weak barbed bisimulation is the same as for the strong version with every occurence of $\xrightarrow{\tau}$ replaced by $\Rightarrow$ and every occurence of $\downarrow_b$ replaced by $\Downarrow_b$. Weak barbed bisimilarity $\dot{\approx}$ and equivalence $\approx$ are then defined analogously to the strong case.

In the above definition of barbed equivalence, we limit ourselves to parallel contexts for two reasons. Firstly, in the pi calculus the use of more general contexts yields a relation that is strictly stronger than labelled bisimilarity (defined below). Secondly, the parallel context is very close to the intuition of an intruder that runs simultaneously with the protocol to be verified. We define the standard notion of (late strong) labelled bisimulation as follows.

Definition 1.4.4 *A binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a labelled bisimulation if whenever $P \mathcal{R} Q$,*

1. *if $P \xrightarrow{\tau} P'$ then there is Q' such that $Q \xrightarrow{\tau} Q'$ and $P' \mathcal{R} Q'$; and*
2. *if $P \xrightarrow{(\nu \tilde{b})\,\overline{a}\,M} P'$ with $\{\tilde{b}\} \cap \mathrm{fn}(Q) = \emptyset$ then there is Q' such that $Q \xrightarrow{(\nu \tilde{b})\,\overline{a}\,M} Q'$ and $P' \mathcal{R} Q'$; and*
3. *if $P \xrightarrow{a(x)} P'$ then there is Q' such that $Q \xrightarrow{a(x)} Q'$ and for all $M \in \mathcal{M}$ $P'\{^M/_x\} \mathcal{R} Q'\{^M/_x\}$; and*
4. *the symmetrical versions of 1,2,3 for transitions of Q hold.*

We let $\sim$ be the union of all labelled bisimulations.

Unlike in the pi calculus, labelled bisimulation is too strong a notion of equivalence for spi processes: It distinguishes between the (barbed equivalent) processes $(\nu k)\,\overline{a}\langle \mathrm{E}_k(M)\rangle$ and $(\nu k)\,\overline{a}\langle \mathrm{E}_k(N)\rangle$ whenever $M \neq N$, which contradicts the modelling assumption of perfect encryption. This problem was adressed using environment-sensitive bisimulation, which we treat in the following chapter.

Chapter 2

Comparing Environment-sensitive Bisimulations for the spi Calculus

The immediate questions on the various competing notions of bisimulation for the spi calculus are (1) how they relate to each other, and (2) how each of them relates to *barbed equivalence*, which is a uniformly defined contextual notion of bisimulation that is usually considered prime among all bisimilarities [MS92]. So far, these questions have only been treated in parts and not always fully correctly.

- Boreale et al. [BDP99] proved that alley bisimilarity is a sound approximation of barbed equivalence. However, as we show in Section 2.1.5, their proof was flawed (although later repaired by Boreale [Bor04]). Moreover, they proved alley bisimilarity complete w.r.t. barbed equivalence for the class of *structurally image-finite* processes, i.e., processes that are image-finite up to structural equivalence.
- Abadi and Gordon [AG98] proved framed bisimilarity sound with respect to barbed equivalence, but were aware that their notion of framed bisimulation is strictly stronger than barbed equivalence. Their conjectured counterexample uses pairing and encryption; we simplify it to only use encryption.
- Elkjær et al. [EHHO99] proposed that fenced bisimulation would coincide with framed bisimulation, but their proof is also flawed and the inclusion only holds in one direction (see Section 2.2.1): framed bisimilarity is not contained in fenced bisimilarity.
- Two different notions of alley bisimilarity were defined by Boreale et al. [BDP99], but the journal version [BDP02] only contains the weaker notion, alley bisimilarity, which is complete with respect to barbed equivalence (see above). The stronger notion, here called trellis bisimilarity, was shown by Frendrup et al. [FHJ01] to coincide with fenced bisimilarity.

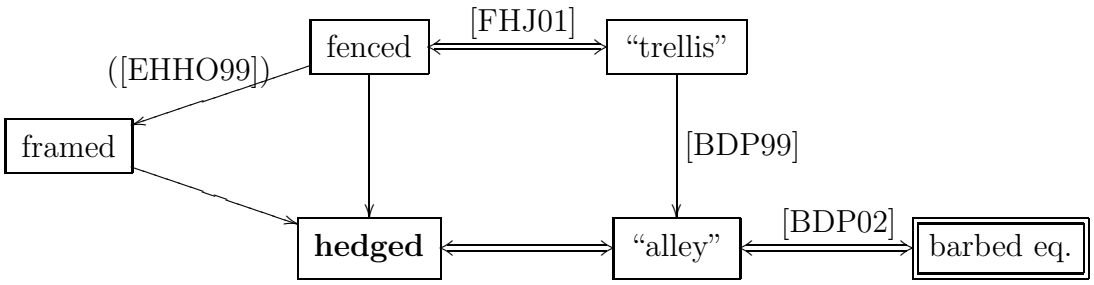

Figure 2.1: Comparing Bisimulations

As in the pi-calculus [MPW92], bisimulations exist in early and late variants, as well as strong and weak. We study the weak early variants of all bisimulations, since the existence of a weak transition may be undecidable [Hüt02]. However, our results applies without modifications to weak and/or late bisimulations.

In Figure 2.1, we pictorially summarize the various relations. Briefly, we introduce a new bisimilarity, called hedged bisimilarity, in the style of framed bisimilarity, and prove it equivalent to alley bisimilarity and thus to barbed equivalence. We then proceed to relate hedged bisimilarity to framed and fenced, showing that fenced is strictly stronger than framed, which itself is strictly stronger than hedged bisimilarity. We highlight the differences between the bisimilarities by means of examples, strengthen them by disproofs within a novel framework for comparison of environment-sensitive bisimilarities, and finally give embeddings of hedged into certain up-to variants of framed and fenced bisimilarity.

Outline

We motivate and define environment-sensitive bisimulations (Section 2.1) and introduce a general framework for comparing environment-sensitive bisimilarities (Section 2.3). All of the relations in Figure 2.1 are supported by proofs for the previously unknown positive results (Section 2.5) and both counterexamples (Section 2.2) and disproofs (Section 2.5.6) for the negative results.

To further clarify the structural differences between the bisimilarities, we describe the bisimilarities as categories (Section 2.6). Then, we attempt to relate these categories in terms of embedding functors and equivalences. To obtain these results, we also need to interpret up-to techniques in this setting.

2.1 Environment-Sensitive Bisimulations

As motivated in the Introduction, bisimulations in the spi calculus must take into account the knowledge of the observing environment—potentially any kind of malicious attacker—at any moment in time. Since the interaction between an environment and a process is fully described by the exchange of messages, it is important to spell out the power of attackers in the spi calculus model. Inspired by [DY83], (1) the environment learns new messages by reading any kind of data that the process sends on public channels; (2) the environment may then independent of any further message exchange update its knowledge by

- detecting whether a known message is a name or an encrypted message,
- using known names to decrypt known messages (usually called *analysis*),
- comparing known messages to other known messages,
- storing known messages for later comparison, encryption or decryption;

(3) the environment may then send on public channels any message that it is able to create (a procedure that is usually called *synthesis*) by using its current knowledge, plus fresh names that it might create itself. Summing up, in environments we need to represent knowledge in such a way that we may at any time calculate from this knowledge those messages that it can currently synthesize according to the above operations.

One straightforward approach is to jointly model the behavior of a pair $e_P \vdash P$, where e_P contains the current knowledge as the list of all messages ever received by the environment from the observed process that is now in state P. Any input action of P is governed by the environment in that we consider inputs for only those messages that can be synthesized by e_P. Any output action of P resulting in P' is used to increase the knowledge of e_P resulting in $e_{P'}$ by simply appending the new message to the list.

An alternative approach is to represent the knowledge "more efficiently" in *irreducible* form by carrying out the full analysis after every process output. An advantage is that the required data structure becomes smaller, and that the synthesis can be carried out directly. Note that we cannot represent the full synthesis, because it usually is infinite.

The mode of observation changes slightly under the regime of environment-sensitive bisimulation. Instead of observing only a single process (as in $e_P \vdash P$), an environment now observes a pair of processes "at the same time" (as in $e_{PQ} \vdash P \mathcal{S} Q$, where $\mathcal{S}$ denotes a bisimulation relation, and where e_{PQ} simply may be a pair (e_P, e_Q) of single environments). The spi calculus principle of distinguishing between *possessing* a message (e.g., $\mathrm{E}_k(a)$) and *knowing* it (e.g., knowing that k is the encryption key, such that a becomes known as well) comes into play again: In standard process calculi it is required that whenever P emits a message M, the message emitted

by the simulating process Q must coincide syntactically with M, whereas the spi calculus must be more permissive.

The definition of bisimulation must permit different messages to be sent to the environment by P and Q, but only under the requirement that they lead to corresponding analyzes in the respective environment component of e_{PQ}. This idea is captured by the notion of *consistency* which guarantees that an environment can not decrypt a message received from P unless it can also decrypt *the corresponding message* received from Q. As a consequence, environments must also properly keep track of the association of the messages received from P and Q.

Depending on the type of the data structure e as introduced below (frame-theory pairs, hedges, or substitution pairs) the notions of analysis, synthesis, and consistency appear in different forms or only implicitly, which renders their comparison non-trivial. Irreducibility may be enforced on the data structure. Synthesis may be expressed explicitly by means of substitution on expressions.

To prove two processes P and Q barbed equivalent, one wants to find a bisimulation $\mathcal{S}$ such that $e \vdash P \,\mathcal{S}\, Q$ for some initial environment e that knows all the free names of both processes. Free names are public, so under a worst case assumption a malicious attacker might take advantage of any and all of them.

We use some meta-variables for denoting bisimilarities and their corresponding environments. Let $\sim_{\mathrm{x}}$ and $\sim_{\mathrm{y}}$ denote bisimilarities with corresponding environments $e_{\mathrm{x}} \in E_{\mathrm{x}}$ and $e_{\mathrm{y}} \in E_{\mathrm{y}}$. Whenever $\mathcal{R} \subseteq \mathbf{E} \times \mathcal{P} \times \mathcal{P}$ is an environment-sensitive relation for some kind of environments $\mathbf{E}$, we define $\mathcal{R}^{-1} := \{\, (e^{-1}, Q, P) \mid (e, P, Q) \in \mathcal{R} \,\}$ for some suitably defined inversed environment e^{-1}. We write that $e \vdash P \,\mathcal{R}\, Q$ if $(e, P, Q) \in \mathcal{R}$, otherwise $e \vdash P \not\mathcal{R}\, Q$. $\mathcal{R}$ is *symmetric* if $\mathcal{R} = \mathcal{R}^{-1}$.

2.1.1 Framed and Fenced Bisimulations

Framed bisimulation [AG98], by Abadi and Gordon, was the first environment-sensitive bisimulation proposed for the spi calculus. The original definition was for a late bisimulation. Here, we study an early variant in order to sharpen the comparison with the bisimulation defined by Boreale et al. [BDP99, BDP02]. Abadi and Gordon also used a different calculus, with a complex set of messages containing integers, pairing and general encryption keys but without general guards, choice and general "let". The examples distinguishing the bisimilarities, that we will exhibit in Section 2.2, are chosen to also be expressible in the original spi calculus [AG99].

In framed bisimulation the environment consists of a frame and a theory. A frame is a set of names known to the environment. A theory is a set of pairs of messages considered equivalent by the environment, e.g., since it does not possess the relevant decryption keys.

Definition 2.1.1 *A* frame *is a finite subset of* $\mathcal{N}$*. A* theory *is a finite subset of*

$\mathcal{M} \times \mathcal{M}$. **FT** *is the set of all frame-theory pairs. If* $B \subset \mathcal{M}$ *is finite then we define* $\mathrm{Id}_B := \{ (b, b) \mid b \in B \}$. *If* th *is a theory, we define* $\mathrm{th}^{-1} := \{ (N, M) \mid (M, N) \in \mathrm{th} \}$, $\pi_1(\mathrm{th}) := \{ M \mid (M, N) \in \mathrm{th} \}$ *and* $\pi_2(\mathrm{th}) := \{ N \mid (M, N) \in \mathrm{th} \}$. *The names of a theory is defined as* $\mathrm{n}(\mathrm{th}) := \mathrm{n}(\pi_1(\mathrm{th}) \cup \pi_2(\mathrm{th}))$.

A frame-theory pair is *consistent* if the theory only contains pairs of encrypted messages that the environment can not decrypt. Moreover, the theory may not relate a given message to two different messages, since this intuitively yields a noticable difference between the two sides .

Definition 2.1.2 *A frame-theory pair* $(\mathrm{fr}, \mathrm{th})$ *is* consistent *iff for all messages* M *and* N *such that* $(M, N) \in \mathrm{th}$ *we have that*

1. $M, N \notin \mathcal{N}$
2. *If* $(M', N') \in \mathrm{th}$ *then* $M = M' \iff N = N'$
3. *If* $M = \mathrm{E}_a(M')$ *and* $N = \mathrm{E}_b(N')$ *then* $\mathrm{fr} \cap \{a, b\} = \emptyset$.

Example 2.1.3 *One consistent and three inconsistent frame-theory pairs:*

$$
\begin{aligned}
(\{a, b\}, \{(\mathrm{E}_k(a), \mathrm{E}_k(a)), (\mathrm{E}_l(a), \mathrm{E}_l(b))\}) &\text{ is consistent.} \\
(\mathrm{fr}, \mathrm{th}) = (\{a, b\}, \{(\mathrm{E}_k(a), k)\}) &\text{ violates condition 1 for consistency.} \\
(\mathrm{fr}', \mathrm{th}') = (\{a, b\}, \{(\mathrm{E}_k(a), \mathrm{E}_b(a))\}) &\text{ violates condition 3 for consistency.} \\
(\mathrm{fr} \cup \mathrm{fr}', \mathrm{th} \cup \mathrm{th}') = (\{a, b\}, \{(\mathrm{E}_k(a), k), (\mathrm{E}_k(a), \mathrm{E}_b(a))\}) &\text{ violates all three} \\
&\text{conditions for consistency.}
\end{aligned}
$$

The *synthesis* $\mathcal{S}(\cdot)$ of a frame-theory pair is the set of message pairs constructed by encrypting message pairs from the theory with keys from the frame. This models the ability of the intruder to construct new messages from previously received ones. The environment considers equivalent any message pair in the synthesis.

Definition 2.1.4 *If* $(\mathrm{fr}, \mathrm{th})$ *is a frame-theory pair, we let* $\mathcal{S}(\mathrm{fr}, \mathrm{th})$ *be the smallest subset of* $\mathcal{M} \times \mathcal{M}$ *containing* $\mathrm{th} \cup \mathrm{Id}_{\mathrm{fr}}$ *and satisfying*

$$
(\textsc{syn-enc}) \ \frac{(M, N) \in \mathcal{S}(\mathrm{fr}, \mathrm{th}) \qquad (a, b) \in \mathcal{S}(\mathrm{fr}, \mathrm{th})}{(\mathrm{E}_a(M), \mathrm{E}_b(N)) \in \mathcal{S}(\mathrm{fr}, \mathrm{th})}
$$

We write $(\mathrm{fr}, \mathrm{th}) \vdash M \leftrightarrow N$ *for* $(M, N) \in \mathcal{S}(\mathrm{fr}, \mathrm{th})$*; otherwise* $(\mathrm{fr}, \mathrm{th}) \vdash M \not\leftrightarrow N$.

To compare the knowledge of environments we use the following pre-order:

Definition 2.1.5 $(\mathrm{fr}, \mathrm{th}) \leq (\mathrm{fr}', \mathrm{th}')$ *iff* $\mathcal{S}(\mathrm{fr}, \mathrm{th}) \subseteq \mathcal{S}(\mathrm{fr}', \mathrm{th}')$. *Two frame-theory pairs* $(\mathrm{fr}, \mathrm{th})$ *and* $(\mathrm{fr}', \mathrm{th}')$ *are* $\mathcal{M}$-equivalent*, written* $(\mathrm{fr}, \mathrm{th}) \geqslant (\mathrm{fr}', \mathrm{th}')$*, when* $\mathcal{S}(\mathrm{fr}, \mathrm{th}) = \mathcal{S}(\mathrm{fr}', \mathrm{th}')$.

A *framed process pair* is a triple $((\mathrm{fr}, \mathrm{th}), P, Q)$ where fr is a frame, th is a theory and P and Q are processes. A *framed relation* $\mathcal{R}$ is a set of framed process pairs. $\mathcal{R}$ is *consistent* if $(\mathrm{fr}, \mathrm{th})$ is consistent whenever $(\mathrm{fr}, \mathrm{th}) \vdash P\ \mathcal{R}\ Q$. Now we have enough notation to define framed bisimilarity.

Definition 2.1.6 *A consistent framed relation $\mathcal{R}$ is a* framed simulation *if whenever*
$(\mathrm{fr}, \mathrm{th}) \vdash P\ \mathcal{R}\ Q$ *we have that*

1. *If $P \xrightarrow{\tau} P'$ then there exists Q' such that $Q \xrightarrow{\tau} Q'$ and $(\mathrm{fr}, \mathrm{th}) \vdash P'\ \mathcal{R}\ Q'$.*
2. *If $P \xrightarrow{a(x)} P'$, $a \in \mathrm{fr}$, $B \subset \mathcal{N}$ is finite, $B \cap (\mathrm{fn}(P,Q) \cup \mathrm{fr} \cup \mathrm{n}(\mathrm{th})) = \emptyset$, $M, N \in \mathcal{M}$, and $(\mathrm{fr} \cup B, \mathrm{th}) \vdash M \leftrightarrow N$,*
   then there exists Q'
   such that $Q \xrightarrow{a(x)} Q'$ and $(\mathrm{fr} \cup B, \mathrm{th}) \vdash P'\{^M/_x\}\ \mathcal{R}\ Q'\{^N/_x\}$
3. *If $P \xrightarrow{(\nu \tilde{c})\,\overline{a}\,M} P'$, $a \in \mathrm{fr}$, and $\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{fr} \cup \mathrm{n}(\pi_1(\mathrm{th}))) = \emptyset$,*
   then there exist $Q', N, \tilde{d}$ with $\{\tilde{d}\} \cap (\mathrm{fn}(Q) \cup \mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th}))) = \emptyset$ and
   (a) *$Q \xrightarrow{(\nu \tilde{d})\,\overline{a}\,N} Q'$, and*
   (b) *there exist $\mathrm{fr}', \mathrm{th}'$ with $(\mathrm{fr}, \mathrm{th}) \leq (\mathrm{fr}', \mathrm{th}')$ and $(\mathrm{fr}', \mathrm{th}') \vdash M \leftrightarrow N$ such that $(\mathrm{fr}', \mathrm{th}') \vdash P'\ \mathcal{R}\ Q'$.*

$\mathcal{R}$ is a framed bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are framed simulations.*

It is worth noting how the new environment after output transitions is characterized: names and message pairs can be freely added as long as the synthesis is extended conservatively and the new output messages are kept indistinguishable. Moreover, since the two messages may differ in structure (as long as this is not noticed by the environment), they may also contain different fresh names.
Since any union of framed bisimulations is a framed bisimulation there exists a greatest framed bisimulation, denoted $\sim_{\mathrm{f}}$, which is the union of all framed bisimulations.

Fenced bisimulation was defined by [EHHO99], who proved it to be a sound and complete approximation to framed bisimulation. (In Section 2.2.1, we show that this is in fact not the case.) The difference between the definitions is that fenced bisimulation replaces the existential quantification over frames and theories in case *3.(b)* of Definition 2.1.6 with a function ξ, defined in Table 2.1, that extends a given frame-theory pair with a new pair of messages. The function ξ works by recursively decomposing the pair of received messages and adding the cores to the environment, verifying consistency whenever anything has been added. Elkjær et al. showed that ξ creates a *minimal* consistent extension whenever there exists one. Since our message grammar is simpler the definition of ξ has been correspondingly simplified.

$\xi(\mathrm{fr}, \mathrm{th}, M, N)$
  IF $((\mathrm{fr}, \mathrm{th}) \vdash M \leftrightarrow N)$ THEN RETURN $(\mathrm{fr}, \mathrm{th})$
  IF $(M = N \in \mathcal{N})$ DO
    $(\mathrm{fr}', \mathrm{th}') := (\mathrm{fr} \cup \{M\}, \mathrm{th})$
    $\lambda := \emptyset$
    FOR EACH $(\mathrm{E}_k(M'), \mathrm{E}_l(N')) \in \mathrm{th}$ DO
      IF $(k = l = M)$ THEN DO
        $\mathrm{th}' := \mathrm{th}' \setminus \{(\mathrm{E}_k(M'), \mathrm{E}_l(N'))\}$
        $\lambda := \lambda \cup \{(M', N')\}$
      DONE
      ELSIF $(k = M \vee l = M)$ THEN RETURN $\perp$
    DONE
    FOR EACH $(M', N') \in \lambda$ DO
      $(\mathrm{fr}', \mathrm{th}') := \xi(\mathrm{fr}', \mathrm{th}', M', N')$
    DONE
    RETURN $(\mathrm{fr}', \mathrm{th}')$
  ELSIF $(M = \mathrm{E}_k(M') \wedge N = \mathrm{E}_l(N'))$ DO
    IF $(k = l \in \mathrm{fr})$ THEN RETURN $\xi(\mathrm{fr}, \mathrm{th}, M', N')$
    IF $(k \in \mathrm{fr} \vee l \in \mathrm{fr})$ THEN RETURN $\perp$
    RETURN $(\mathrm{fr}, \mathrm{th} \cup \{(M, N)\})$
  DONE
  RETURN $\perp$
DONE

Table 2.1: An Algorithm for the Function ξ

Definition 2.1.7 *A consistent framed relation $\mathcal{R}$ is a* fenced simulation *if whenever* $(\mathrm{fr}, \mathrm{th}) \vdash P \mathcal{R} Q$ *we have that*

1. *If* $P \xrightarrow{\tau} P'$
   then there exists Q' *such that*
   $Q \xrightarrow{\tau} Q'$ *and* $(\mathrm{fr}, \mathrm{th}) \vdash P' \mathcal{R} Q'$.
2. *If* $P \xrightarrow{a(x)} P'$, $a \in \mathrm{fr}$, $B \subset \mathcal{N}$ *is finite,* $B \cap (\mathrm{fn}(P, Q) \cup \mathrm{fr} \cup \mathrm{n}(\mathrm{th})) = \emptyset$, $M, N \in \mathcal{M}$, *and* $(\mathrm{fr} \cup B, \mathrm{th}) \vdash M \leftrightarrow N$,
   then there exists Q' *such that*
   $Q \xrightarrow{a(x)} Q'$ *and* $(\mathrm{fr} \cup B, \mathrm{th}) \vdash P'\{^M/_x\} \mathcal{R} Q'\{^N/_x\}$
3. *If* $P \xrightarrow{(\nu \tilde{c})\, \overline{a}\, M} P'$, $a \in \mathrm{fr}$ *and* $\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{fr} \cup \mathrm{n}(\pi_1(\mathrm{th}))) = \emptyset$
   there exist $Q', N, \tilde{d}$ *with* $\{\tilde{d}\} \cap (\mathrm{fn}(Q) \cup \mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th}))) = \emptyset$
   such that $Q \xrightarrow{(\nu \tilde{d})\, \overline{a}\, N} Q'$ *and* $\xi(\mathrm{fr}, \mathrm{th}, M, N) \vdash P' \mathcal{R} Q'$.

$\mathcal{R}$ is a fenced bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are fenced simulations.*

Since any union of fenced bisimulations is a fenced bisimulation there exists a greatest fenced bisimulation, denoted $\sim_{\#}$, which is the union of all fenced bisimulations.

2.1.2 Alley and Trellis Bisimulations

Boreale et al. [BDP99, BDP02] defined environment-sensitive semantics and a corresponding weak bisimulation for the spi calculus. As mentioned earlier, we call their bisimulation "alley" in order to distinguish it from the other bisimulations, which are also environment-sensitive. The authors proved that alley bisimulation is a sound approximation of barbed equivalence, and that the approximation is complete for the class of "structurally image-finite" processes. They also studied a number of "up-to" techniques for this bisimulation. In this work, we study a strong version of this bisimulation, in order to simplify the setting for our comparison.

Formally, Boreale et al. defined two levels of operational semantics, one for the behavior of processes, and another one for the corresponding behavior of environments. We adapt their formalism to the style without per-process environments, following [FHJ01], by simply incorporating the environment semantics rules into the definition of bisimulation. In alley bisimulation, the environment is a pair of substitutions. We denote by **SS** the set of all alleys, i.e., the set of all substitution pairs.

Any set of messages, e.g., the messages in the codomain of a substitution, might be reduced via decryption using the notion of analysis (cf. [Pau98]).

Definition 2.1.8 *The* analysis $\mathcal{A}(S)$ *and the* irreducibles $\mathcal{I}(S)$ *of a set $S \subseteq \mathcal{M}$ are defined as follows: $\mathcal{A}(S)$ is the smallest subset of $\mathcal{M}$ containing S and satisfying*

$$(\text{SET-DEC})\ \frac{\mathrm{E}_a(M) \in \mathcal{A}(S) \quad a \in \mathcal{A}(S)}{M \in \mathcal{A}(S)}$$

and $\mathcal{I}(S) := \mathcal{A}(S) \setminus \{\, \mathrm{E}_a(M) \mid a \in \mathcal{A}(S) \,\}$.

The function $\mathrm{core}_\sigma(M)$ decrypts a message M as far as possible, i.e., peels out the *core* of M using the knowledge of a substitution σ. We use the shorthands $\mathcal{I}(\sigma)$ for $\mathcal{I}(\mathrm{range}(\sigma))$ and $\mathcal{A}(\sigma)$ for $\mathcal{A}(\mathrm{range}(\sigma))$. We define

$$\mathrm{core}_\sigma(M) \stackrel{\text{def}}{=} \begin{cases} \mathrm{core}_\sigma(M') & \text{if } M = \mathrm{E}_a(M') \text{ and } a \in \mathcal{I}(\sigma) \\ M & \text{otherwise} \end{cases}$$

Thus, we can decompose any message M into $\mathrm{E}_{b_n}(\cdots \mathrm{E}_{b_2}(\mathrm{E}_{b_1}(\mathrm{core}_\sigma(M)))\cdots)$ for any substitution σ with $\{b_1 \ldots, b_n\} \subseteq \mathcal{I}(\sigma)$; if $\mathrm{core}_\sigma(M) = \mathrm{E}_a(N)$, then $a \notin \mathcal{I}(\sigma)$. Writing $\mathcal{C}(\sigma, x) := \mathrm{core}_\sigma(\sigma(x))$, we get $\mathcal{I}(\sigma) = \{\mathcal{C}(\sigma, x) \mid x \in \mathrm{dom}(\sigma)\}$.

Boreale et al. define a substitution pair to be consistent if the two substitutions have the same domain and enable the same guards, i.e., (σ, ρ) is consistent if $\mathrm{dom}(\sigma) = \mathrm{dom}(\rho)$ and $\forall \phi \in \mathcal{G}$ such that $\mathrm{n}(\phi) \subseteq \mathrm{dom}(\sigma)$ we have $[\![\phi\sigma]\!] = [\![\phi\rho]\!]$. They also give an alternative characterization, for this choice of guard and message language, that avoids the infinite quantification over guards above. In this paper, we work with this alternative characterization, stating that a pair of substitutions is consistent if they allow us to decrypt corresponding messages in precisely corresponding ways. In other words, it does not simply suffice to decrypt to corresponding cores, but we must also use corresponding keys for the decryption.

Definition 2.1.9 *A pair of substitutions (σ, ρ) is* consistent*, written $\sigma \cong \rho$, iff σ and ρ have the same domain $\{x_1 \ldots, x_n\}$ and the following conditions hold:*

1. $\mathcal{C}(\sigma, x_i) \in \mathcal{N} \iff \mathcal{C}(\rho, x_i) \in \mathcal{N}$
2. $\mathcal{C}(\sigma, x_i) = \mathcal{C}(\sigma, x_j) \iff \mathcal{C}(\rho, x_i) = \mathcal{C}(\rho, x_j)$
3. *For each $i \in \{1, 2, \ldots, n\}$ there is a tuple $\tilde{\iota} = \iota_1, \ldots, \iota_m$ such that*

$$\begin{aligned} \sigma(x_i) &= \mathrm{E}_{\mathcal{C}(\sigma, x_{\iota_m})}(\cdots \mathrm{E}_{\mathcal{C}(\sigma, x_{\iota_2})}(\mathrm{E}_{\mathcal{C}(\sigma, x_{\iota_1})}(\mathcal{C}(\sigma, x_i)))\cdots) \\ \rho(x_i) &= \mathrm{E}_{\mathcal{C}(\rho, x_{\iota_m})}(\cdots \mathrm{E}_{\mathcal{C}(\rho, x_{\iota_2})}(\mathrm{E}_{\mathcal{C}(\rho, x_{\iota_1})}(\mathcal{C}(\rho, x_i)))\cdots) \end{aligned}$$

Example 2.1.10 *To illustrate this definition, we let*

$$\begin{aligned} \sigma &= \left\{{}^{a}/_{x_1}\right\}\left\{{}^{\mathrm{E}_a(b)}/_{x_2}\right\}\left\{{}^{\mathrm{E}_k(\mathrm{E}_a(c))}/_{x_3}\right\} \\ \rho &= \left\{{}^{a}/_{x_1}\right\}\left\{{}^{\mathrm{E}_a(b)}/_{x_2}\right\}\left\{{}^{\mathrm{E}_k(\mathrm{E}_k(d))}/_{x_3}\right\} \\ \tau &= \left\{{}^{a}/_{x_1}\right\}\left\{{}^{\mathrm{E}_a(c)}/_{x_2}\right\}\left\{{}^{\mathrm{E}_k(\mathrm{E}_k(c))}/_{x_3}\right\} \end{aligned}$$

Then we have that $\sigma \cong \rho$, $\sigma \cong \tau$ and $\rho \cong \tau$. Note that we must allow two different names (b and c) to correspond, in order to relate σ and τ. If both σ and ρ acquire knowledge of the key (name) k we get that $\sigma\left\{{}^{k}/_{y}\right\} \ncong \rho\left\{{}^{k}/_{y}\right\}$, since they violate condition 3 of $\cong$ by using different encryption keys in the decryption of the third message. We also have that $\rho\left\{{}^{k}/_{y}\right\} \ncong \tau\left\{{}^{k}/_{y}\right\}$, since they violate condition 2 by having c in $\mathcal{I}(\tau)$ correspond to both b and d in $\mathcal{I}(\rho)$.

We also define a notion of the synthesis of a consistent substitution pair.

Definition 2.1.11 *If $\sigma \cong \rho$ we write $(\sigma, \rho) \vdash M \leftrightarrow N$ iff there is G such that $\mathrm{n}(G) \subseteq \mathrm{dom}(\sigma)$, $\mathbf{e}(G\sigma) = M$ and $\mathbf{e}(G\rho) = N$. The synthesis of a consistent pair of substitutions is defined as $\mathcal{S}(\sigma, \rho) := \{(M, N) \mid (\sigma, \rho) \vdash M \leftrightarrow N\}$.*

An *alley relation* $\mathcal{R}$ is a set of triples $((\sigma,\rho),P,Q)$ with $\mathrm{dom}(\sigma)=\mathrm{dom}(\rho)$. $\mathcal{R}$ is *consistent* if $(\sigma,\rho)\vdash P\,\mathcal{R}\,Q$ implies that $\sigma\cong\rho$.

Definition 2.1.12 *A consistent alley relation $\mathcal{R}$ is an* alley simulation *if whenever $(\sigma,\rho)\vdash P\,\mathcal{R}\,Q$ the following conditions hold:*

1. *If $P\xrightarrow{\tau}P'$ then there exists Q' such that $Q\xrightarrow{\tau}Q'$ and $(\sigma,\rho)\vdash P'\,\mathcal{R}\,Q'$.*
2. *If $P\xrightarrow{a(x)}P'$ and there are $M,G,\tilde{b},b$ such that $\mathbf{e}(G\sigma)=M$, $(\sigma,\rho)\vdash a\leftrightarrow b$, $\{\tilde{b}\}=\mathrm{n}(G)$ and $\tilde{b}\cap\mathrm{fn}(P,Q,\rho,\sigma)=\emptyset$, then there exist $\tilde{y},Q'$ with $\tilde{y}\subset\mathcal{V}$, $|\tilde{y}|=|\tilde{b}|$, $\tilde{c}\cap\mathrm{dom}(\sigma)=\emptyset$ such that $Q\xrightarrow{b(x)}Q'$ and $(\sigma\{{}^{\tilde{b}}/_{\tilde{y}}\},\rho\{{}^{\tilde{b}}/_{\tilde{y}}\})\vdash P'\{{}^{\mathbf{e}(G\sigma)}/_{x}\}\,\mathcal{R}\,Q'\{{}^{\mathbf{e}(G\rho)}/_{x}\}$.*
3. *If $P\xrightarrow{(\nu\tilde{c})\,\overline{a}\,M}P'$ with $\mathrm{fn}(P,\sigma)\cap\{\tilde{c}\}=\emptyset$, $(\sigma,\rho)\vdash a\leftrightarrow b$ and $x\notin\mathrm{dom}(\sigma)$ then there are $Q',N,\tilde{d}$ with $\mathrm{fn}(Q,\rho)\cap\{\tilde{d}\}=\emptyset$ such that $Q\xrightarrow{(\nu\tilde{d})\,\overline{b}\,N}Q'$ and $(\sigma\{{}^{M}/_{x}\},\rho\{{}^{N}/_{x}\})\vdash P'\,\mathcal{R}\,Q'$.*

$\mathcal{R}$ is an alley bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are alley simulations.*

Note the difference with respect to the previous bisimulations. Here, the environment is extended simply by mechanically adding the new messages (for output) or the new names (for input), without reducing the environment at all. This also gives a minimal extension (cf. fenced bisimulation) since no extra information may be added. Since we use substitutions as environments, consistency is vital. Otherwise, the creation of message pairs by applying both substitutions to the same formula gives meaningless results.

Trellis bisimulation is a strengthened variant of alley bisimulation, studied (not under this name) by [BDP99]. There, two different notions of consistency of environments were proposed, of which one was rejected since it was considered too strong. This rejected notion, here callede *strong consistency*, constitutes the basis for trellis bisimulation.

Definition 2.1.13 *A consistent pair of substitutions $\sigma\cong\rho$ is strongly consistent, written $\sigma\cong_{\mathrm{s}}\rho$, if $\mathcal{C}(\sigma,x)\in\mathcal{N}$ implies that $\mathcal{C}(\sigma,x)=\mathcal{C}(\rho,x)$.*

This resembles the definition of consistency of frame-theory pairs in that two different names may never be considered equal. Strong consistency has a corresponding bisimulation, called *trellis* in this paper, that was defined and compared to fenced bisimulation by [FHJ01]. We recapitulate and strengthen the results of the comparison in Section 2.5.

Definition 2.1.14 *An alley relation $\mathcal{R}$ is strongly consistent if $(\sigma,\rho)\vdash P\,\mathcal{R}\,Q$ implies $\sigma\cong_{\mathrm{s}}\rho$. We call* trellis bisimulation *a strongly consistent alley bisimulation.*

Since any union of alley/trellis bisimulations is an alley/trellis bisimulation itself there exists a greatest alley/trellis bisimulation, denoted $\sim_{\mathrm{a}}/\sim_{\mathrm{s}}$, which is the union of all alley/trellis bisimulations.

2.1.3 Hedged Bisimulation

Hedged bisimulation is introduced in this paper in order to clarify the differences between framed, fenced, and alley bisimulation. Recall that alley bisimulation, unlike its counterparts, does not force two processes to always send the same names; it rather remembers that the respective names correspond to each other. The basic idea of hedges is to mimic the lack of correspondence in frame-theory pairs by dropping the separate frame component, but to include corresponding names as part of the theory. The resulting theory is then called a *hedge*.

Definition 2.1.15 *A* hedge *is a theory, i.e., a finite subset of $\mathcal{M} \times \mathcal{M}$. We denote by* $\mathbf{H}$ *the set of all hedges. The* synthesis $\mathcal{S}(\cdot)$ *of a hedge h is defined as the smallest subset of $\mathcal{M} \times \mathcal{M}$ containing h and satisfying*

$$(\text{SYN-ENC})\ \frac{(M,N) \in \mathcal{S}(h) \qquad (a,b) \in \mathcal{S}(h)}{(\mathrm{E}_a(M), \mathrm{E}_b(N)) \in \mathcal{S}(h)}$$

We write $h \vdash M \leftrightarrow N$ for $(M,N) \in \mathcal{S}(h)$, $h \vdash M \not\leftrightarrow N$ otherwise.

A hedge is consistent if the hedge only contains pairs of names and pairs of encrypted messages that can not be decrypted by the environment. We also require that no message is considered to be equivalent to two different messages.

Definition 2.1.16 *A hedge h is* consistent *iff whenever $(M,N) \in h$*

1. *$M \in \mathcal{N} \iff N \in \mathcal{N}$*
2. *If $(M',N') \in h$ then $M = M' \iff N = N'$*
3. *If $M = \mathrm{E}_a(M')$ and $N = \mathrm{E}_b(N')$ then $a \notin \pi_1(h)$ and $b \notin \pi_2(h)$.*

Example 2.1.17 *A consistent hedge and four inconsistent hedges:*

$g = \{(a,a),(b,c),(c,k),(\mathrm{E}_k(b),\mathrm{E}_l(a))\}$ *is consistent.*
$h_1 = \{(a,\mathrm{E}_k(a)),(b,c),(\mathrm{E}_k(b),\mathrm{E}_l(a))\}$ *violates condition 1 for consistency.*
$h_2 = \{(a,c),(b,c),(\mathrm{E}_k(b),\mathrm{E}_l(a))\}$ *violates condition 2 for consistency.*
$h_3 = \{(a,a),(k,c),(\mathrm{E}_k(b),\mathrm{E}_l(a))\}$ *violates condition 3 for consistency.*

Note that $h_1 \cup h_2 \cup h_3$ violates all three conditions. Indeed, if a hedge h is not consistent then $h \cup h'$ is not consistent for any hedge h'.

The difference between a consistent hedge and a consistent frame-theory pair is that we do not require that the hedge receives the same names from both processes, so they do not need to use the same channels and encryption keys. We defined hedge consistency in this fashion in order to yield a closer correspondence to the state of affairs in the alley style of bisimulation.

The difference between a consistent hedge and a consistent substitution pair is that the former is minimal (cf. Lemma 2.4.11) since it only contains undecryptable messages (i.e., cores) and that no duplicate message pairs are allowed. The third condition for consistent substitutions (Definition 2.1.9) roughly corresponds to the definition of hedge analysis (cf. Lemma 2.4.29) .

Definition 2.1.18 *The* analysis $\mathcal{A}(h)$ *and the* irreducibles $\mathcal{I}(h)$ *of a hedge* h *are defined as follows:* $\mathcal{A}(h)$ *is the smallest subset of* $\mathcal{M}\times\mathcal{M}$ *containing* h *and satisfying*

$$(\textsc{hedge-dec})\ \frac{(\mathrm{E}_a(M),\mathrm{E}_b(N))\in\mathcal{A}(h)\qquad (a,b)\in\mathcal{A}(h)}{(M,N)\in\mathcal{A}(h)}$$

and $\mathcal{I}(h) \stackrel{\text{def}}{=} \mathcal{A}(h)\setminus\{\,(\mathrm{E}_a(M),\mathrm{E}_b(N)) \mid (a,b)\in\mathcal{A}(h)\wedge M,N\in\mathcal{M}\,\}$.

The analysis of hedges decrypts pairs of messages using pairs of names that are considered equivalent by the environment. The resulting notion of irreducibles corresponds to the result of the ξ-function of fenced bisimulation (cf. Lemma 2.4.27).

Now that the environment and notions of consistency are defined, the definition of hedged bisimulation is straightforward. A *hedged relation* $\mathcal{R}$ is a subset of $\mathbf{H}\times\mathcal{P}\times\mathcal{P}$. We say that $\mathcal{R}$ is *consistent* if $h\vdash P\,\mathcal{R}\,Q$ implies that h is consistent.

Definition 2.1.19 *A consistent hedged relation* $\mathcal{R}$ *is a* hedged simulation *if whenever* $h\vdash P\,\mathcal{R}\,Q$ *we have that*

1. *If* $P\xrightarrow{\tau}P'$ *then there exists* Q' *such that* $Q\xrightarrow{\tau}Q'$ *and* $h\vdash P'\,\mathcal{R}\,Q'$.
2. *If* $P\xrightarrow{a(x)}P'$, $h\vdash a\leftrightarrow b$, $B\subset\mathcal{N}$ *is finite,* $B\cap(\mathrm{fn}(P,Q)\cup\mathrm{n}(h))=\emptyset$, $M,N\in\mathcal{M}$, *and* $h\cup\mathrm{Id}_B\vdash M\leftrightarrow N$, *then there exists* Q' *such that* $Q\xrightarrow{b(x)}Q'$ *and* $h\cup\mathrm{Id}_B\vdash P'\{^M/_x\}\,\mathcal{R}\,Q'\{^N/_x\}$.
3. *If* $P\xrightarrow{(\nu\tilde{c})\,\overline{a}\,M}P'$, $h\vdash a\leftrightarrow b$ *and* $\{\tilde{c}\}\cap(\mathrm{fn}(P)\cup\mathrm{n}(\pi_1(h)))=\emptyset$ *there exist* $Q',N,\tilde{d}$ *with* $\{\tilde{d}\}\cap(\mathrm{fn}(Q)\cup\mathrm{n}(\pi_2(h)))=\emptyset$ *such that* $Q\xrightarrow{(\nu\tilde{d})\,\overline{b}\,N}Q'$ *and* $\mathcal{I}(h\cup\{(M,N)\})\vdash P'\,\mathcal{R}\,Q'$.

$\mathcal{R}$ *is a* hedged bisimulation *if both* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are hedged simulations.*

On process output we use $\mathcal{I}(\cdot)$ to construct the new hedge after the transition. This entails applying all decryptions that the environment can do, producing—as in

fenced bisimulation—the minimal extension of the hedge h with (M, N). As in the other bisimulations, this extension may turn out to be inconsistent, signifying that the hedge has detected a difference between the messages received from the process pair.

Since any union of hedged bisimulations is a hedged bisimulation there exists a greatest hedged bisimulation, denoted $\sim_{\mathrm{h}}$, which is the union of all hedged bisimulations.

2.1.4 Weak Bisimulation

All of the above bisimilarities also exist in weak versions. The definitions are identical, apart from the substitution of $\cdot \stackrel{\cdot}{\Longrightarrow} \cdot$ as defined below for any occurrences of $\cdot \stackrel{\cdot}{\rightarrow} \cdot$. The results comparing the bisimilarities all hold for both the strong and the weak versions, with the single exception of Proposition 2.3.5 which only applies to weak bisimilarities.

Definition 2.1.20 *We let* $\stackrel{\tau}{\Longrightarrow} \stackrel{\text{def}}{=} (\stackrel{\tau}{\rightarrow})^*$ *and* $\stackrel{\mu}{\Longrightarrow} \stackrel{\text{def}}{=} \stackrel{\tau}{\Longrightarrow}\stackrel{\mu}{\rightarrow}\stackrel{\tau}{\Longrightarrow}$ *if* $\mu \neq \tau$.

2.1.5 Up-to techniques

Generally, in bisimulation proofs, one needs to exhibit that two related terms $P\ \mathcal{S}\ Q$ possess matching transitions that allow them to proceed to terms $P'\ \mathcal{S}\ Q'$ that are again related by the same relation $\mathcal{S}$. For this to hold, the relation $\mathcal{S}$ is often required to be large or even infinite. So-called "up-to techniques" are an effective way of reducing the size of the relation $\mathcal{S}$ to be exhibited to prove two processes bisimilar: the obligation to proceed to related terms $P'\ \mathcal{S}\ Q'$ is relaxed by requiring only $P'\ \mathcal{F}(\mathcal{S})\ Q'$ for some $\mathcal{F}(\mathcal{S})$ (typically, $\mathcal{S} \subset \mathcal{F}(\mathcal{S})$). For example, for ordinary (i.e., non environment-sensitive) bisimulations, these techniques include "bisimulation up to strong bisimilarity" for CCS by Milner [Mil89], where derivatives only need to be related by $\sim\mathcal{S}\sim$. An up-to technique is *sound* for a certain bisimilarity, if one can prove that whenever two terms are related using this up-to technique, then they are also bisimilar. Sangiorgi [San98] introduced the concept of *respectful* up-to techniques, and proved them sound and composable.

In the context of environment-sensitive relations, up-to techniques naturally have to take into account the additional effects on the environment component. Moreover, there are techniques (as we will see below) that exclusively affect the environment component.

Boreale et al. [BDP02] defined several up-to techniques for alley bisimulation. Since these up-to techniques were later used as the basis for a proof system [BG02], it is interesting to study them also for other bisimilarities. In the remainder of this section, we define and discuss up-to techniques for alley and hedged bisimulation;

adapting the latter definition to framed and fenced is straightforward. In Section 2.2 we then show some cases where up-to techniques are not sound for framed and fenced bisimilarity. For alley bisimulation, we define up to forgetfulness (originally called weakening; we reserve that term for a more general notion), contraction, additional[1] restriction, and injective renaming, closely following Boreale et al. [BDP02].

Definition 2.1.21 *If $\mathcal{R}$ is an alley relation, we define $\mathcal{R}_{\mathrm{t}}$ for $\mathrm{t} \in \{\mathrm{f}, \mathrm{c}, \mathrm{r}, \mathrm{i}\}$ as the smallest alley relation containing $\mathcal{R}$ and satisfying the following rules:*

- up to forgetfulness*:*
$$\frac{(\sigma\{{}^{M}\!/_{x}\}, \rho\{{}^{N}\!/_{x}\}) \vdash P\ \mathcal{R}_{\mathrm{f}}\ Q}{(\sigma, \rho) \vdash P\ \mathcal{R}_{\mathrm{f}}\ Q}$$
- up to contraction*:*
$$\frac{(\sigma, \rho) \vdash P\ \mathcal{R}_{\mathrm{c}}\ Q}{(\sigma\{{}^{M}\!/_{x}\}, \rho\{{}^{N}\!/_{x}\}) \vdash P\ \mathcal{R}_{\mathrm{c}}\ Q} \quad \begin{array}{l}\text{if there is } F:\ M = \mathbf{e}(F\sigma),\ N = \mathbf{e}(F\rho) \\ \text{and } \mathrm{n}(F) \cap \mathrm{fn}(\sigma, \rho, P, Q) = \emptyset\end{array}$$
- up to additional restriction*:*
$$\frac{(\sigma, \rho) \vdash P\ \mathcal{R}_{\mathrm{r}}\ Q}{(\sigma, \rho) \vdash (\nu n)\, P\ \mathcal{R}_{\mathrm{r}}\ Q} \text{ if } n \notin \mathrm{fn}(\sigma) \qquad \frac{(\sigma, \rho) \vdash P\ \mathcal{R}_{\mathrm{r}}\ Q}{(\sigma, \rho) \vdash P\ \mathcal{R}_{\mathrm{r}}\ (\nu n)\, Q} \text{ if } n \notin \mathrm{fn}(\rho)$$
- up to injective renaming*: We let* $\sigma@\{{}^{M_1}\!/_{x_1}, \ldots, {}^{M_n}\!/_{x_n}\} := \{{}^{M_1\sigma}\!/_{x_1}, \ldots, {}^{M_n\sigma}\!/_{x_n}\}$. *Then*
$$\frac{(\sigma, \rho) \vdash P\ \mathcal{R}_{\mathrm{i}}\ Q}{(\sigma'@\sigma, \rho'@\rho) \vdash P\sigma'\ \mathcal{R}_{\mathrm{i}}\ Q\rho'} \text{ if } \sigma', \rho' : \mathcal{N} \to \mathcal{N} \text{ are injective.}$$

*We use words (or tuples) $\tilde{\mathrm{t}} \in \{\mathrm{f}, \mathrm{c}, \mathrm{r}, \mathrm{i}\}^*$ (with the single letters $\{\mathrm{f}, \mathrm{c}, \mathrm{r}, \mathrm{i}\}$ abbreviating the respective $\{f$*orgetfulness*, c*ontraction*, additional *r*estriction*, i*njective renaming$\}$ to denote sequential composition of up-to techniques by $\mathcal{R}_{\epsilon} := \mathcal{R}$ and $\mathcal{R}_{\tilde{\mathrm{t}}\cdot\mathrm{u}} := (\mathcal{R}_{\tilde{\mathrm{t}}})_{\mathrm{u}}$.*

The concept of alley bisimulation up to $\tilde{\mathrm{t}}$ *is defined in close analogy to Definition 2.1.12 as follows: all bisimulation clauses of this definition require the derivatives P', Q' to satisfy $(\sigma', \rho') \vdash P'\ \mathcal{R}\ Q'$ for some σ', ρ' depending on the current clause; for the up-to variant we replace those conditions by the weaker $(\sigma', \rho') \vdash P'\ \mathcal{R}_{\tilde{\mathrm{t}}}\ Q'$.*

We also define up-to techniques for hedged bisimulation.

[1] We call this technique "up to *additional* restriction" to distinguish it from a different technique called "up to restriction" [MPW92].

Definition 2.1.22 *We write $h(\sigma, \rho)$ for $\{(M\sigma, N\rho) \mid (M, N) \in h\}$.*

If $\mathcal{R}$ is a hedged relation, we define $\mathcal{R}_{\mathrm{t}}$ for $\mathrm{t} \in \{\mathrm{w}, \mathrm{r}, \mathrm{b}, \mathrm{s}\}$ as the smallest hedged relation containing $\mathcal{R}$ and satisfying the following rules:

- up to weakening
$$\frac{h \vdash P \; \mathcal{R} \; Q}{\mathcal{I}(h') \vdash P \; \mathcal{R}_{\mathrm{w}} \; Q} \text{ if } \mathcal{S}(h') \subseteq \mathcal{S}(h)$$
- up to additional restriction*:*
$$\frac{h \vdash P \; \mathcal{R}_{\mathrm{r}} \; Q}{h \vdash (\nu n)\, P \; \mathcal{R}_{\mathrm{r}} \; Q} \text{ if } n \notin \mathrm{n}(\pi_1(h)) \qquad \frac{h \vdash P \; \mathcal{R}_{\mathrm{r}} \; Q}{h \vdash P \; \mathcal{R}_{\mathrm{r}} \; (\nu n)\, Q} \text{ if } n \notin \mathrm{n}(\pi_2(h))$$
- up to bijective renaming
$$\frac{h \vdash P \; \mathcal{R} \; Q}{h(\sigma, \rho) \vdash P\sigma \; \mathcal{R}_{\mathrm{b}} \; Q\rho} \text{ if } \sigma, \rho : \mathcal{N} \to \mathcal{N} \text{ are bijective}$$
- up to labelled bisimulation
$$\frac{h \vdash P \; \mathcal{R} \; Q}{h \vdash P' \; \mathcal{R}_{\mathrm{l}} \; Q'} \text{ if } P \sim P' \text{ and } Q \sim Q'$$

Note that $\mathcal{R}_{\mathrm{bb}} = \mathcal{R}_{\mathrm{b}}$, $\mathcal{R}_{\mathrm{ww}} = \mathcal{R}_{\mathrm{w}}$ and $\mathcal{R}_{\mathrm{ll}} = \mathcal{R}_{\mathrm{l}}$, so we can take $h \vdash P \; \mathcal{R} \; Q$ (rather than $h \vdash P \; \mathcal{R}_{\mathrm{t}} \; Q$) as precondition in the corresponding rules. Moreover, we clearly have that $\mathcal{R} \subseteq \mathcal{R}_{\mathrm{b}}$, $\mathcal{R} \subseteq \mathcal{R}_{\mathrm{w}}$, and $\mathcal{R}_{\mathrm{bw}} = \mathcal{R}_{\mathrm{wb}}$.

A consistent symmetric hedged relation is a hedged bisimulation up to $\tilde{\mathrm{t}} \in \{\mathrm{w}, \mathrm{r}, \mathrm{b}, \mathrm{s}\}^*$ if it satisfies the definition of hedged bisimulation (Definition 2.1.19) with the condition $h' \vdash P' \; \mathcal{R} \; Q'$ on the derivatives replaced by the weaker $h' \vdash P' \; \mathcal{R}_{\tilde{\mathrm{t}}} \; Q'$.

The definitions of framed/fenced bisimulation up to restriction or weakening are the same as for hedged bisimulation above. Note the definition of weakening; we permit an arbitrary reduction of the synthesis, and then apply $\mathcal{I}(\cdot)$ to ensure that we end up with a minimal environment. (For frame-theory pairs, one instead recursively applies the ξ function of fenced bisimilarity, starting with an empty theory.)

For environment-sensitive relations, "up to weakening permits discarding environment entries" [BDP02], thus it denotes the *reduction* of environment knowledge. This is in contrast to the field of type systems, where the notion of weakening usually denotes the *addition* of potentially unnecessary information to the (typing) environment (cf. up to contraction). In this paper, we reserve the term weakening for arbitrary reduction of environment knowledge, while forgetfulness denotes the discarding of individual environment entries. Clearly, weakening is the more general notion.

In order for an up-to technique to be usable, it must at least be *sound*.

Definition 2.1.23 *An up-to technique* $\tilde{\mathrm{t}}$ *is* sound *for* $\sim_{\mathrm{h}}$ *(*$\sim_{\mathrm{a}}$*) if every hedged (alley) bisimulation up to* $\tilde{\mathrm{t}}$ *is contained in* $\sim_{\mathrm{h}}$ *(*$\sim_{\mathrm{a}}$*).*

The following properties prove useful for the analysis of up-to techniques.

Definition 2.1.24 *An up-to technique* $\tilde{\mathrm{t}}$ *is an* expansion *if we always have* $\mathcal{R} \subseteq \mathcal{R}_{\tilde{\mathrm{t}}}$*, and* monotonous *(with respect to* $\subseteq$*) if* $\mathcal{S} \subseteq \mathcal{R}$ *implies that* $\mathcal{S}_{\tilde{\mathrm{t}}} \subseteq \mathcal{R}_{\tilde{\mathrm{t}}}$*.*

Note that all of the up-to techniques defined above are monotonous expansions.

Major parts of the proof system of Boreale & Gorla [BG02] arise from the following proposition. We also use its contrapositive in Section 2.2 to disprove the soundness of certain up-to techniques.

Proposition 2.1.25 *If an up-to technique* $\tilde{\mathrm{t}}$ *is a monotonous expansion and sound for* $\sim_{\mathrm{x}}$*, then* $e \vdash P \sim_{\mathrm{x}} Q$ *and* $(e', P', Q') \in \{(e, P, Q)\}_{\tilde{\mathrm{t}}}$ *imply that* $e' \vdash P' \sim_{\mathrm{x}} Q'$*.*

The soundness of an up-to technique $\tilde{\mathrm{t}}$ is usually proved by showing $\mathcal{R}_{\tilde{\mathrm{t}}}$ to be a bisimulation whenever $\mathcal{R}$ is a bisimulation up to $\tilde{\mathrm{t}}$. Another way is to use the following property.

Proposition 2.1.26 *If* $\tilde{u}$ *is an expansion and* $\tilde{t}\tilde{u}$ *is sound then* $\tilde{t}$ *is also sound.*

As an example of where the usual soundness proof schema does not work, Boreale et al. assert ([BDP02], p. 982, proof of Proposition B.5 (4.17)) that $\mathcal{R}_{\mathrm{f}}$ is "straightforwardly" an alley bisimulation whenever $\mathcal{R}$ is an alley bisimulation up to forgetfulness. However, the assertion is false[2]; as we will see this is due to a subtle issue regarding name freshness, similarly to the examples on framed and fenced bisimilarity in Section 2.2. We here exhibit an alley bisimulation up to forgetfulness $\mathcal{R}$ such that $\mathcal{R}_{\mathrm{f}}$ is not an alley bisimulation.

Example 2.1.27 *Fix* a *and let* $P := (\nu l)\,\overline{a}\langle l\rangle.\,\mathbf{0}$*. All transitions of* P *are of the form* $P \xrightarrow{(\nu l)\,\overline{a}\,l} \mathbf{0}$ *with* $l \neq a$*.*

Take fixed k, x, y *such that* $k \neq a$ *and* $x \neq y$*, and let* $\sigma := \{{}^{a}/_{x}\}\{{}^{\mathrm{E}_k(a)}/_{y}\}$ *and* $\mathcal{R} := \{((\sigma, \sigma), P, P)\} \cup \{\,((\sigma\{{}^{u}/_{t}\}, \sigma\{{}^{u}/_{t}\}), \mathbf{0}, \mathbf{0}) \mid t \neq x, y \wedge u \neq a, k\,\}$*. It is easy to verify that* $\mathcal{R}$ *is an alley bisimulation. Thus, since forgetfulness is an expansion,* $\mathcal{R}$ *is also an alley bisimulation up to forgetfulness.*

We show that $\mathcal{R}_{\mathrm{f}}$ *is not an alley bisimulation. We have* $(\{{}^{a}/_{x}\}, \{{}^{a}/_{x}\}) \vdash P\ \mathcal{R}_{\mathrm{f}}\ P$*,* $y \notin \mathrm{dom}(\{{}^{a}/_{x}\})$ *and* $P \xrightarrow{(\nu k)\,\overline{a}\,k} \mathbf{0}$*. This transition needs to be simulated by* P*, resulting in an alley process pair in* $\mathcal{S} := \{((\{{}^{a}/_{x}\}\{{}^{k}/_{y}\}, \{{}^{a}/_{x}\}\{{}^{u}/_{y}\}), \mathbf{0}, \mathbf{0}) \mid u \neq a\}$*. This simulated transition is problematic in two independent ways.*

[2]The reader familiar with the notion of respectful up-to techniques defined by [San98] will note that this also implies that up to forgetfulness is not respectful, in contrast to what was asserted by [BDP02] (p. 964, footnote 3).

1. *Name-clash in the range: k occurs as a bound output, but was free in the original environment σ. Thus, all transitions $P \xrightarrow{(\nu l)\,\overline{a}\,l} \mathbf{0}$ that needed to be simulated under the original environment would have $l \neq k$. The possible resulting environments after such a transition then only contains $\{{}^{l}/_{z}\}$ for $l \neq k$, and forgetfulness alone does not let us rename l to k.*
2. *Name-clash in the domain: The output message k was substituted for a variable y already used in the original environment. Clearly, there is no way to turn an environment in $\mathcal{R}$ (where $\{{}^{\mathrm{E}_k(a)}/_{y}\}$) into an environment of $\mathcal{S}$ (where $\{{}^{k}/_{y}\}$) by mere forgetfulness.*

Summing up, we have $\mathcal{S} \cap \mathcal{R}_{\mathrm{f}} = \emptyset$, so $\mathcal{R}_{\mathrm{f}}$ is not an alley bisimulation.

Example 2.1.27 does not disprove the soundness of alley bisimilarity up to forgetfulness, because $\mathcal{R}_{\mathrm{f}} \subset \sim_{\mathrm{a}}$ still holds for the $\mathcal{R}$ of the example. We have only showed that the "standard" way of proving the soundness of an up-to technique does not work for this case. As an aside, one of the main results of [BDP02], the proof of the soundness of alley bisimilarity with respect to barbed equivalence, depends on the soundness of forgetfulness (p. 983, proof of Proposition B.5 (4.17), *Up to parallel composition*).

Noting that the problems in Example 2.1.27 were due to name clashes in both the range and the domain of the environment substitutions, we proposed to use injective renaming to solve the first problem and introduced a new up-to technique called domain renaming to deal with the latter.

Definition 2.1.28 *If $\mathcal{R}$ is an alley relation, we define $\mathcal{R}_{\mathrm{d}}$ as the smallest alley relation containing $\mathcal{R}$ and satisfying the following rule:*

- up to domain renaming

$$\frac{(\sigma, \rho) \vdash P\ \mathcal{R}_{\mathrm{d}}\ Q}{(\beta\sigma, \beta\rho) \vdash P\ \mathcal{R}_{\mathrm{d}}\ Q} \text{ if } \beta : \mathcal{N} \rightharpoonup \mathrm{dom}(\sigma) \text{ is injective.}$$

We define the concept of alley bisimulation up to $\tilde{\mathrm{t}} \in \{\mathrm{f}, \mathrm{c}, \mathrm{r}, \mathrm{i}, \mathrm{d}\}^*$ as in Definition 2.1.21.

Indeed, following this proposal Boreale [Bor04] repaired the proof of soundness of alley bisimilarity up to forgetfulness, using the up-to technique of domain renaming defined above. The new result is that if $\mathcal{R}$ is an alley bisimulation up to forgetfulness, additional restriction and structural equivalence (not defined in this thesis) then $\mathcal{R}_{\mathrm{frs}}$ is an alley bisimulation up to injective renaming and variable renaming, where the latter up-to techniques are also shown to be sound and composable. Thus, using Proposition 2.1.26 we get that up to forgetfulness is sound for alley bisimilarity.

In order to relate framed and hedged bisimilarity, we need to prove the soundness of hedged bisimilarity up to weakening. In doing the proof, we encounter the same problems as seen in Example 2.1.27, but they are easier to deal with due to the simpler structure of hedges. To wit, Theorem 2.5.17 will show that if $\mathcal{R}$ is a hedged bisimulation up to weakening and bijective renaming then $\mathcal{R}_{\mathrm{wb}}$ is a hedged bisimulation, i.e., hedged bisimulation is sound up to restriction and/or bijective renaming.

2.2 Distinguishing Examples

In this section we exhibit differences between framed, fenced and hedged bisimulation. In Section 2.5 we show that hedged and alley bisimulation are equivalent, so these examples also distinguish alley bisimulation from framed and fenced.

2.2.1 Fenced vs. Framed and Hedged — Counting Extruded Names

The following example distinguishes fenced bisimulation from its competitors due to a subtle requirement concerning the data involved in the simulation of output transitions, especially the conditions on the choice of bound names:

$$\begin{array}{lllll} P & := & (\nu nkl) & \overline{a}\langle \mathrm{E}_l(\mathrm{E}_k(n))\rangle.P' & \qquad P' := (\nu m)\,\overline{a}\langle m\rangle.\,\mathbf{0} \\ Q & := & (\nu nk) & \overline{a}\langle \mathrm{E}_k(n)\rangle.Q' & \qquad Q' := (\nu m)\,\overline{a}\langle m\rangle.\,\mathbf{0} \end{array}$$

Although there is no reason for P and Q to be distinguished, fenced bisimulation does so because it insists that outputs of fresh names be simulated without renaming.

We write $\mathrm{pwd}(\tilde{n})$ to denote that $\tilde{n}$ is a tuple of *p*air*w*ise *d*ifferent names.

Proposition 2.2.1 $(\{a\}, \emptyset) \vdash P \not\sim_{\#} Q$.

Proof. The transitions of P are $P \xrightarrow{(\nu nkl)\,\overline{a}\ \mathrm{E}_l(\mathrm{E}_k(n))} P'$ where $\mathrm{pwd}(n,k,l,a)$. The transitions of Q are of the form $Q \xrightarrow{(\nu n'k')\,\overline{a}\ \mathrm{E}_{k'}(n')} Q'$ where $\mathrm{pwd}(n',k',a)$. We then get the theory $\{(\mathrm{E}_l(\mathrm{E}_k(n)), \mathrm{E}_{k'}(n'))\}$. Since n,k,l are pairwise different, there must be a name $z \in \{n,k,l\} \setminus \{n',k'\}$.

Now P' must simulate the transition $Q' \xrightarrow{(\nu z)\,\overline{a}\,z} \mathbf{0}$. We have $P' \xrightarrow{(\nu m)\,\overline{a}\,m} \mathbf{0}$ for all $m \neq a$. For the resulting frame-theory pair to be consistent, we must have $m = z$, but since $z \in \mathrm{n}(\pi_1(\mathrm{th}))$ this transition can not be used. □

Fenced bisimulation fails since it cannot simulate an output of a particular bound name when this name is already known by the simulating environment. This problem could in all likelyhood be fixed at the level of the definition of the bisimilarity, at the cost of increased discrepancy to the other definitions.

On the other hand, both framed and hedged bisimulation succeed in relating the two processes, but in different ways.

Proposition 2.2.2 $\{(a,a)\} \vdash P \sim_{\mathrm{h}} Q$.

Proof. A hedged bisimulation relating P and Q is given by

$$\begin{array}{rcl}
\mathcal{R} & := & \{\ (\{(a,a)\}, P, Q)\ \} \\
& \cup & \{\ (\{(a,a),(\mathrm{E}_l(\mathrm{E}_k(n)),\mathrm{E}_l(n))\}, P', Q') \mid \mathrm{pwd}(a,k,l,n)\ \} \\
& \cup & \{\ (\{(a,a),(\mathrm{E}_l(\mathrm{E}_k(n)),\mathrm{E}_l(n)),(m,m)\}, \mathbf{0}, \mathbf{0}) \mid \mathrm{pwd}(a,k,l,n,m)\ \} \\
& \cup & \{\ (\{(a,a),(\mathrm{E}_l(\mathrm{E}_k(n)),\mathrm{E}_l(n)),(w,k)\}, \mathbf{0}, \mathbf{0}) \mid \mathrm{pwd}(a,k,l,n,w)\ \}
\end{array}$$

□

Note the addition of (w,k) to the hedge, simply denoting that the environment cannot distinguish the two different names.

Proposition 2.2.3 $(\{a\}, \emptyset) \vdash P \sim_{\mathrm{f}} Q$.

Proof. A framed bisimulation relating P and Q is given by

$$\begin{array}{l}
\mathcal{R} := \{((\{a\}, \emptyset), P, Q)\} \\
\quad \cup \{((\{a,k\}, \{(\mathrm{E}_l(\mathrm{E}_k(n)),\mathrm{E}_l(n))\}), P', Q') \mid \mathrm{pwd}(a,k,l,n)\} \\
\quad \cup \{((\{a,k,m\}, \{(\mathrm{E}_l(\mathrm{E}_k(n)),\mathrm{E}_l(n))\}), \mathbf{0}, \mathbf{0}) \mid \mathrm{pwd}(a,k,l,m,n)\}
\end{array}$$

□

Note the addition of k to the frame, which relieves the process P' from simulating the critical bound output of z by Q' (see the previous proof). The name created by Q' must be different from the names in the current frame and theory, so by simply adding k to the frame—which is allowed in framed bisimulation, but due to the minimality property not in fenced bisimulation—this name must be chosen different from k, and thus P' may also create it.

In comparison, we may conclude that $\sim_{\mathrm{f}}$ equates the processes through a non-minimal extension of the frame (which could be considered "cheating"), while $\sim_{\mathrm{h}}$ equates them (more adequately) through correspondence.

Corollary 2.2.4 $\sim_{\mathrm{f}}$ *is not a subset of* $\sim_{\#}$.

An Up-To Interpretation.

The examples above shows that removing a piece of information (the name k) from a frame-theory pair actually may increase its power to distinguish between processes. [BDP02] call this removal of knowledge from the environment *weakening*, which intuitively should be sound with respect to a bisimulation: An environment with more information at hand has more possibilities to discover a difference between two processes. While Corollary 2.5.18 will state that hedged bisimulation is sound up to weakening, the above examples show that this is not the case for framed and fenced bisimulation.

Proposition 2.2.5 *Framed and fenced bisimulations are not sound up to weakening.*

Proof. As seen above, $(\{a\}, \{(\mathrm{E}_l(\mathrm{E}_k(n)), \mathrm{E}_l(n))\}) \vdash P' \not\sim_{\#} Q'$. Similarly to Proposition 2.2.3, we also have $(\{a,k\}, \{(\mathrm{E}_l(\mathrm{E}_k(n)), \mathrm{E}_l(n))\}) \vdash P' \sim_{\#} Q'$. Clearly, $((\{a\}, \{(\mathrm{E}_l(\mathrm{E}_k(n)), \mathrm{E}_l(n))\}), P', Q') \in$
$\{((\{a,k\}, \{(\mathrm{E}_l(\mathrm{E}_k(n)), \mathrm{E}_l(n))\}), P', Q')\}_{\mathrm{w}}$. Since up to weakening is a monotonous expansion, Proposition 2.1.25 then gives that it is not sound with respect to fenced bisimilarity. The same reasoning holds for $\sim_{\mathrm{f}}$. □

2.2.2 Framed vs. Hedged — Unknown Names Must Not Matter

This example is a version without pairing of the example by [AG98] showing that framed bisimilarity is not complete w.r.t. barbed equivalence. We define

$$\begin{array}{lllll} P & := & (\nu klm) & \overline{a}\langle \mathrm{E}_k(\mathrm{E}_l(m))\rangle. & (\quad \overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0} \quad) \\ Q & := & (\nu kn) & \overline{a}\langle \mathrm{E}_k(n)\rangle. & \overline{a}\langle n\rangle.\,\mathbf{0} \end{array}$$

We show that $\{(a,a)\} \vdash P \sim_{\mathrm{h}} Q$ and that $(\{a\}, \emptyset) \vdash P \not\sim_{\mathrm{f}} Q$. Intuitively, this means that for $\sim_{\mathrm{f}}$ the identity of the unknown name matters, although an attacker doesn't have any means to verify this identity.

Proposition 2.2.6 $\{(a,a)\} \vdash P \sim_{\mathrm{h}} Q$.

Proof. We show that the relation

$$\begin{aligned} \mathcal{R} := & \{\, (\{(a,a)\}, P, Q) \,\} \\ & \cup \{\, (h(k,l,m), (\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0}), \overline{a}\langle n\rangle.\,\mathbf{0}) \mid k,l,m,n \in \mathcal{N} \setminus \{a\} \,\} \\ & \cup \{\, (h(k,l,m,n) \cup \{(m,n)\}, \mathbf{0}, \mathbf{0}) \mid k,l,m,n \in \mathcal{N} \setminus \{a\} \,\} \\ & \cup \{\, (h(k,l,m,n) \cup \{(l,n)\}, \mathbf{0}, \mathbf{0}) \mid k,l,m,n \in \mathcal{N} \setminus \{a\} \,\} \end{aligned}$$

where $h(k,l,m,n) := \{(a,a),(\mathrm{E}_k(\mathrm{E}_l(m)),\mathrm{E}_k(n))\}$ and k,l,m,n are pairwise different wherever they occur, is a hedged bisimulation. $\mathcal{R}$ is "trivially" consistent as we never receive a nor the outermost keys of encrypted messages.

- As $P \xrightarrow{(\nu k,l,m)\,\overline{a}\ \mathrm{E}_k(\mathrm{E}_l(m))} (\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0})$ whenever k,l,m and a are pairwise different, we seek $Q', N, \tilde{d}$ with $Q \xrightarrow{(\nu\tilde{d})\,\overline{a}\,N} Q', a \notin \{\tilde{d}\}$ and $\mathcal{I}(\{(a,a),(\mathrm{E}_k(\mathrm{E}_l(m)),N)\}) \vdash (\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0})\ \mathcal{R}\ Q'$.
  We may choose $\tilde{d} = (k,n)$, $N = \mathrm{E}_k(n)$ and $Q' = \overline{a}\langle n\rangle.\,\mathbf{0}$. We also have that $\mathcal{I}(h(k,l,m,n)) = h(k,l,m,n)$.
- The output transitions of Q can be handled in the same way.
- The transition $(\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0}) \xrightarrow{\overline{a}\,m} \mathbf{0}$ can be simulated by $\overline{a}\langle n\rangle.\,\mathbf{0} \xrightarrow{\overline{a}\,n} \mathbf{0}$.
- The transition $(\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0}) \xrightarrow{\overline{a}\,l} \mathbf{0}$ can also be simulated by $\overline{a}\langle n\rangle.\,\mathbf{0} \xrightarrow{\overline{a}\,n} \mathbf{0}$.
- The transition $\overline{a}\langle n\rangle.\,\mathbf{0} \xrightarrow{\overline{a}\,n} \mathbf{0}$ can be simulated by $(\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0}) \xrightarrow{\overline{a}\,m} \mathbf{0}$.

The character of hedges is visible in the addition of the pair (l,n) resp. (l,n) in the two last rows of $\mathcal{R}$ above; l and m are both different from n, but the hedge notes that they should correspond. □

Proposition 2.2.7 $(\{a\},\emptyset) \vdash P \not\sim_{\mathrm{f}} Q$.

Proof. By contradiction: assume that $(\{a\},\emptyset) \vdash P \sim_{\mathrm{f}} Q$.

- As a is in the frame and $P \xrightarrow{(\nu k,l,m)\,\overline{a}\ \mathrm{E}_k(\mathrm{E}_l(m))} (\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0})$ whenever k,l,m and a are pairwise different, we need to check that for all such $k,l,m \neq a$ there exist $Q', N, \tilde{d}, \mathrm{fr}, \mathrm{th}$ such that $Q \xrightarrow{(\nu\tilde{d})\,\overline{a}\,N} Q', a \notin \{\tilde{d}\}$, $a \in \mathrm{fr}$ and $(\mathrm{fr},\mathrm{th}) \vdash (\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0}) \sim_{\mathrm{f}} Q'$.
  Any transition of Q is of the form $Q \xrightarrow{(\nu k',n)\,\overline{a}\ \mathrm{E}_{k'}(n)} \overline{a}\langle n\rangle.\,\mathbf{0}$.
- Since $(\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0}) \xrightarrow{\overline{a}\,l} \mathbf{0}$ we need to check that $\overline{a}\langle n\rangle.\,\mathbf{0}$ can simulate the transition. As the only possibility is $\overline{a}\langle n\rangle.\,\mathbf{0} \xrightarrow{\overline{a}\,n} \mathbf{0}$ there is a consistent frame-theory pair $(\mathrm{fr}_1,\mathrm{th}_1)$ such that $(\mathrm{fr}_1,\mathrm{th}_1) \vdash l \leftrightarrow n$, which can only be the case if $l = n \in \mathrm{fr}_1$.
- Since $(\overline{a}\langle m\rangle.\,\mathbf{0} + \overline{a}\langle l\rangle.\,\mathbf{0}) \xrightarrow{\overline{a}\,m} \mathbf{0}$ we need to check that $\overline{a}\langle n\rangle.\,\mathbf{0}$ can simulate the transition. As the only possibility is $\overline{a}\langle n\rangle.\,\mathbf{0} \xrightarrow{\overline{a}\,n} \mathbf{0}$ there is a consistent frame-theory pair $\mathrm{fr}_2, \mathrm{th}_2$ such that $(\mathrm{fr}_2,\mathrm{th}_2) \vdash m \leftrightarrow n$, which can only be the case if $m = n \in \mathrm{fr}_2$.
- We thus have that $m = l$, which is a contradiction.

□

As noted by Abadi and Gordon, framed (and fenced) bisimulation requires us to decide whether to identify n with l or with m at the first output rather than the second, with no possibility of backtracking on the choice.

Note that P contains nondeterministic choice, a construct that is not present in the original spi calculus. However, this example works equally well if we replace the nondeterminism as expressed by choice either by an input and two mutually exclusive guards, as in

$$a(x).(\,[\,x = a\,]\,\overline{a}\langle m\rangle.\,\mathbf{0}\mid[\,x = b\,]\,\overline{a}\langle l\rangle.\,\mathbf{0}\,),$$

or with communication over a private channel, as in

$$(\nu c)\,(\,\overline{c}\langle a\rangle.\,\mathbf{0}\mid c(x).\overline{a}\langle m\rangle.\,\mathbf{0}\mid c(x).\overline{a}\langle l\rangle.\,\mathbf{0}\,).$$

The details are left as an exercise to the reader.

2.2.3 Framed vs. Hedged — Encryption Should Be Perfect

The following example exhibits a striking deficiency of framed bisimulation that is remedied by hedged bisimulation: a frame-theory pair can distinguish between the plaintext of an encrypted message (n in the example) and another random piece of data (m in the example).

$$\begin{array}{lcllcll} P & := & (\nu k, n)\,\overline{a}\langle \mathrm{E}_k(n)\rangle.P' & P' & := & (\nu m)\,\overline{a}\langle m\rangle. & \mathbf{0} \\ Q & := & (\nu k, n)\,\overline{a}\langle \mathrm{E}_k(n)\rangle.Q' & Q' & := & \overline{a}\langle n\rangle. & \mathbf{0} \qquad \text{where } n \neq a. \end{array}$$

We show that $\{(a,a)\} \vdash P \sim_{\mathrm{h}} Q$ and $(\{a\},\emptyset) \vdash P \not\sim_{\mathrm{f}} Q$. We first study the same relations for just the processes P' and Q'. (Note that$\mathrm{fn}(P,Q) \subseteq \mathrm{n}(\{(a,a)\})$, although $\mathrm{fn}(Q') \not\subseteq \mathrm{n}(\{(a,a)\})$.)

Proposition 2.2.8 $\{(a,a)\} \vdash P' \sim_{\mathrm{h}} Q'$.

Proof. We show that the relation

$$\mathcal{R} := \{(\{(a,a)\}, P', Q')\} \cup \{(\{(a,a),(m,n)\}, \mathbf{0}, \mathbf{0}) \mid m \in \mathcal{N} \setminus \{a\}\}$$

is a hedged bisimulation. $\mathcal{R}$ is consistent, since for all hedges h such that $h \in \pi_1(\mathcal{R})$ conditions 1 and 3 are trivially satisfied and condition 2 follows from $n \neq a \neq m$. As neither P' nor Q' do input or internal computation we only have to check the conditions for output. As the output is on a pair of names known to the environment we need to check whether the environment accepts the input.

- As $P' \xrightarrow{(\nu m)\,\overline{a}\,m} \mathbf{0}$ when $m \neq a$, we seek $Q'', N, \tilde{d}$ with $Q' \xrightarrow{(\nu \tilde{d})\,\overline{a}\,N} Q''$, $\{a, n\} \cap \{\tilde{d}\} = \emptyset$ and $\mathcal{I}(\{(a,a),(m,N)\}) \vdash \mathbf{0}\ \mathcal{R}\ Q''$. Clearly, $\tilde{d}$ is empty, $N = n$ and $Q'' = \mathbf{0}$. Trivially $\mathcal{I}(\{(a,a),(m,n)\}) = \{(a,a),(m,n)\}$. Finally, $\{(a,a),(m,n)\} \vdash \mathbf{0}\ \mathcal{R}\ \mathbf{0}$.
- As $Q' \xrightarrow{\overline{a}\,n} \mathbf{0}$, we need to check that there are $P'', N, \tilde{d}$ with $P \xrightarrow{(\nu \tilde{d})\,\overline{a}\,N} P''$, $a \notin \{\tilde{d}\}$ and $\mathcal{I}(\{(a,a),(n,N)\}) \vdash P''\ \mathcal{R}\ \mathbf{0}$. We may choose $\tilde{d} = n$, $N = n$ and $P'' = \mathbf{0}$. Clearly, by inspection, $\mathcal{I}(\{(a,a),(n,n)\}) \vdash \mathbf{0}\ \mathcal{R}\ \mathbf{0}$.

□

The result can be strengthened as follows:

Proposition 2.2.9 *Whenever h is a consistent hedge such that $(a,a) \in h$, $n \notin \pi_2(h)$ and $\pi_2(h) \cap \{\mathrm{E}_n(M) \mid M \in \mathcal{M}\} = \emptyset$ we have that $h \vdash P' \sim_{\mathrm{h}} Q'$.*

Proof. The previous proof also holds for

$$\mathcal{R} := \{(h, P', Q')\} \cup \{\, (h \cup \{(m,n)\}, \mathbf{0}, \mathbf{0}) \mid m \in \mathcal{N} \setminus \mathrm{n}(\pi_1(h)) \,\}$$

since $\mathcal{R}$ is a consistent hedged relation by the preconditions. □

Note that in the above $\mathcal{R}$ the names m and n just correspond. They are previously unknown, can not be used as decryption keys, and there is no way to distinguish between them through further interaction with the process pair.

Since framed bisimilarity does not allow two different names to simply correspond, we have the following negative result.

Proposition 2.2.10 *There is no $(\mathrm{fr}, \mathrm{th})$ such that $a \in \mathrm{fr}$ and $(\mathrm{fr}, \mathrm{th}) \vdash P' \sim_{\mathrm{f}} Q'$.*

Proof. Assume the opposite, and fix $m \neq n$ such that $m \in \mathcal{N} \setminus (\mathrm{fr} \cup \mathrm{n}(\pi_1(\mathrm{th})))$. As $a \in \mathrm{fr}$ and $P \xrightarrow{(\nu m)\,\overline{a}\,m} \mathbf{0}$ there must exist $Q'', N, \tilde{d}, \mathrm{fr}', \mathrm{th}'$ such that $Q' \xrightarrow{(\nu \tilde{d})\,\overline{a}\,N} Q''$, $(\mathrm{fr}', \mathrm{th}')$ is consistent, and $(\mathrm{fr}', \mathrm{th}') \vdash m \leftrightarrow N$. As the only transition of Q' is $Q' \xrightarrow{\overline{a}\,n} \mathbf{0}$ we have that $N = n$. Clearly SYN-ENC (cf. Def. 2.1.4) can not derive $(\mathrm{fr}', \mathrm{th}') \vdash m \leftrightarrow n$. Since $(\mathrm{fr}', \mathrm{th}')$ is consistent, which implies that $(m,n) \notin \mathrm{th}'$, we must have that $m = n \in \mathrm{fr}'$, which is a contradiction. □

Now, we can use the results for P' and Q' to derive results for P and Q.

Proposition 2.2.11 $\{(a,a)\} \vdash P \sim_{\mathrm{h}} Q$

Proof. As (a,a) is in the hedge, P and Q can perform matching output steps. By Proposition 2.2.9, any resulting hedged process pair is in $\sim_{\mathrm{h}}$. □

Proposition 2.2.12 $(\{a\}, \emptyset) \vdash P \not\sim_{\mathrm{f}} Q$

Proof. As a is in the frame, P and Q can perform matching output steps. By Proposition 2.2.10, no resulting framed process pair is in $\sim_{\mathrm{f}}$. □

An Up-To Interpretation.

According to [BDP02], alley bisimulation is sound up to (additional) restriction, meaning that names that are not present in the knowledge of the environment can be restricted in one or both of the processes. By the above example this does not hold for framed bisimulation.

Proposition 2.2.13 *Framed bisimulation is not sound up to additional restriction.*

Proof. As Proposition 2.2.5. □

2.3 Intermezzo

Having seen some examples that distinguish the bisimilarities of Section 2.1, we will now introduce a general framework for relating environment-sensitive bisimilarities.

As a starting point, let us recall the work of [MPW93], comparing early and late bisimilarity (here denoted by $\sim_{\text{early}}$ and $\sim_{\text{late}}$) for the π-calculus. Since π bisimilarities are simply relations on $\mathcal{P}$, mere set inclusion is good enough: $\sim_{\text{early}} \subsetneq \sim_{\text{late}}$.

When moving to environment-sensitive bisimilarities, we must also properly treat the environments. Since the environments of framed and fenced bisimilarity are of the same type, the comparison of the bisimilarities by [EHHO99] could still be done in terms of set inclusion. This is no longer possible when comparing framed/fenced to their hedged and alley counterparts; we must find a more general way to compare environment-sensitive bisimilarities.

2.3.1 Comparing Environment-Sensitive Bisimilarities

Every environment — independently of the kind of data structure involved — straightforwardly induces a binary relation on processes. E.g., to a frame-theory pair (fr, th) we associate the set $\{(P,Q) \mid (\text{fr}, \text{th}) \vdash P \sim_{\text{f}} Q\}$. More generally, we let $\mathcal{R}^{e_{\text{x}}}$ be the set $\{(P,Q) \mid e_{\text{x}} \vdash P \sim_{\text{x}} Q\}$. We then call an environment e_{y} sound with respect to e_{x} if e_{y} does not relate any processes not related by e_{x}, i.e., if the set-theoretic inclusion $\mathcal{R}^{e_{\text{y}}} \subseteq \mathcal{R}^{e_{\text{x}}}$ holds.

It does not make much sense to compare $\mathcal{R}^{e_{\text{y}}}$ and $\mathcal{R}^{e_{\text{x}}}$ for unrelated e_{y} and e_{x}. However, if $\mathbf{E}_{\text{x}}$ and $\mathbf{E}_{\text{y}}$ are the sets of environments of the bisimilarities $\sim_{\text{x}}$ and $\sim_{\text{y}}$, then every function $g : \mathbf{E}_{\text{x}} \to \mathbf{E}_{\text{y}}$ gives us an embedding of the environments of $\sim_{\text{x}}$ into those of $\sim_{\text{y}}$. Such an embedding g is *sound* (point-wise), if $\mathcal{R}^{g(e_{\text{x}})} \subseteq \mathcal{R}^{e_{\text{x}}}$ for all $e_{\text{x}} \in \mathbf{E}_{\text{x}}$; and we call it *complete* (point-wise), if $\mathcal{R}^{g(e_{\text{x}})} \supseteq \mathcal{R}^{e_{\text{x}}}$ for all $e_{\text{x}} \in \mathbf{E}_{\text{x}}$.

Along the same lines, we first considered to call such an embedding g a (point-wise) bisimilarity equivalence if $\mathcal{R}^{g(e_{\text{x}})} = \mathcal{R}^{e_{\text{x}}}$ for all $e_{\text{x}} \in \mathbf{E}_{\text{x}}$. However, this definition

does not yield symmetry: there may be environments in $\mathbf{E}_\mathrm{y}$ that have no equivalent counterpart in $\mathbf{E}_\mathrm{x}$.

Instead, if $\mathcal{R}^{g(e_\mathrm{x})} = \mathcal{R}^{e_\mathrm{x}}$ for all $e_\mathrm{x} \in \mathbf{E}_\mathrm{x}$ we call g a *full abstraction*: It leaves unchanged the set of process pairs related by the environments, which we consider to be the essential property of an environment. Here the full abstraction relation is parameterized with a function, so it is not simply a preorder and thus the standard definition of kernel ($\ker(\leq) := \leq \cap (\leq^{-1})$) is not applicable. However, if $g : \mathbf{E}_\mathrm{x} \to \mathbf{E}_\mathrm{y}$ and $h : \mathbf{E}_\mathrm{y} \to \mathbf{E}_\mathrm{x}$ are full abstractions then (g, h) is a bisimilarity *equivalence* relating $\sim_\mathrm{x}$ and $\sim_\mathrm{y}$. To summarize, we have the following definitions.

Definition 2.3.1 *Assume that $\sim_\mathrm{x}$ and $\sim_\mathrm{y}$ are environment-sensitive bisimilarities, where $\mathbf{E}_\mathrm{x}$ and $\mathbf{E}_\mathrm{y}$ denote the sets of environments of $\sim_\mathrm{x}$ and $\sim_\mathrm{y}$, respectively. Then we define the following relations between $\sim_\mathrm{x}$ and $\sim_\mathrm{y}$.*

Soundness: *$\sim_\mathrm{y}$ is* g-sound *w.r.t. $\sim_\mathrm{x}$*
if $g : \mathbf{E}_\mathrm{x} \to \mathbf{E}_\mathrm{y}$ is such that $\forall e, P, Q :\ g(e) \vdash P \sim_\mathrm{y} Q$ implies $e \vdash P \sim_\mathrm{x} Q$.

Completeness: *$\sim_\mathrm{y}$ is* g-complete *w.r.t. $\sim_\mathrm{x}$*
if $g : \mathbf{E}_\mathrm{x} \to \mathbf{E}_\mathrm{y}$ is such that $\forall e, P, Q :\ e \vdash P \sim_\mathrm{x} Q$ implies $g(e) \vdash P \sim_\mathrm{y} Q$.

Full abstraction: *$\sim_\mathrm{y}$ is* fully g-abstract *w.r.t. $\sim_\mathrm{x}$*
if $g : \mathbf{E}_\mathrm{x} \to \mathbf{E}_\mathrm{y}$ is such that $\forall e, P, Q :\ e \vdash P \sim_\mathrm{x} Q$ iff $g(e) \vdash P \sim_\mathrm{y} Q$.

Equivalence: *$\sim_\mathrm{x}$ and $\sim_\mathrm{y}$ are* (g, h)-equivalent
if $\sim_\mathrm{x}$ is fully g-abstract w.r.t. $\sim_\mathrm{y}$ and $\sim_\mathrm{y}$ is fully h-abstract w.r.t. $\sim_\mathrm{x}$.

Proposition 2.3.2 *Soundness, completeness and full abstraction are reflexive and transitive, in the following sense:*

- *$\sim_\mathrm{x}$ is $\mathrm{Id}_{\mathbf{E}_\mathrm{x}}$-sound (complete, fully abstract) w.r.t. itself.*
- *If $g : \mathbf{E}_\mathrm{x} \to \mathbf{E}_\mathrm{y}$ and $h : \mathbf{E}_\mathrm{y} \to \mathbf{E}_\mathrm{z}$ are such that $\sim_\mathrm{y}$ is g-sound (complete, fully abstract) w.r.t. $\sim_\mathrm{x}$ and $\sim_\mathrm{z}$ is h-sound (complete, fully abstract) with respect to $\sim_\mathrm{y}$, then $\sim_\mathrm{z}$ is $(h \circ g)$-sound (complete, fully abstract) w.r.t. $\sim_\mathrm{x}$.*

Bisimilarity equivalence is reflexive, symmetric and transitive.

- *$\sim_\mathrm{x}$ is $(\mathrm{Id}_{\mathbf{E}_\mathrm{x}}, \mathrm{Id}_{\mathbf{E}_\mathrm{x}})$-equivalent to itself.*
- *If $\sim_\mathrm{x}$ and $\sim_\mathrm{y}$ are (g, h)-equivalent,*
  then $\sim_\mathrm{y}$ and $\sim_\mathrm{x}$ are (h, g)-equivalent.
- *If $\sim_\mathrm{x}$ and $\sim_\mathrm{y}$ are (g, h)-equivalent and $\sim_\mathrm{y}$ and $\sim_\mathrm{z}$ are (g', h')-equivalent,*
  then $\sim_\mathrm{x}$ and $\sim_\mathrm{z}$ are $(g \circ g', h' \circ h)$-equivalent.

Above, a bisimilarity equivalence is constructed from two a priori unrelated full abstractions g and h. Further constraints, such as requiring g and h to be inverse to each other, could be added to make the relation stronger. However, adding constraints on the functions causes problems for the robustness and transitivity of

bisimilarity equivalence. It is not reasonable to require g and h to be inverses, since this can be broken by merely adding a "behavioral copy" of an environment to one of the bisimilarities. A weaker variant of this constraint is to require idempotence of $g \circ h$ and/or $h \circ g$, but it causes problems for transitivity — assuming that $g \circ h$ and $g' \circ h'$ are idempotent, it may well be that $g \circ g' \circ h' \circ h$ is not. (A concrete example is left as an exercise to the reader.)

2.3.2 Examples: Blindness and Inconsistency

We now give some simple examples of soundness, completeness and full abstraction between environment-sensitive bisimilarities.

As it turns out, the definitions of g-soundness and g-completeness as defined in Definition 2.3.1 can often be satisfied by trivial environment mappings. For completeness, the mapping between environments might collapse all environments of $\mathbf{E}_\mathrm{x}$ to a single trivial environment in $\mathbf{E}_\mathrm{y}$ that is equating any pair of processes, being "blind" for any possible distinction. For soundness we have the dual, namely that all environments of $\mathbf{E}_\mathrm{x}$ may be mapped to an environment in $\mathbf{E}_\mathrm{y}$ that discriminates between all process pairs.

Definition 2.3.3 *An environment b is $\sim_\mathrm{y}$-*blind *if $b \vdash P \sim_\mathrm{y} Q$ for all processes P and Q.*

Proposition 2.3.4 *If there is a $\sim_\mathrm{y}$-blind environment b_y and $B_\mathrm{y} : \mathbf{E}_\mathrm{x} \to \mathbf{E}_\mathrm{y}$ has* $\mathrm{range}(B_\mathrm{y}) = \{b_\mathrm{y}\}$ *then $\sim_\mathrm{y}$ is B_y-complete w.r.t. $\sim_\mathrm{x}$.*

Proof. Whenever $e_\mathrm{x} \vdash P \sim_\mathrm{x} Q$ we have $B_\mathrm{y}(e_\mathrm{x}) \vdash P \sim_\mathrm{y} Q$. □

Proposition 2.3.5 *The weak versions of the previously defined bisimilarities all have blind environments:*

1. *$(\emptyset, \emptyset)$ is a blind consistent frame-theory pair.*
2. *$(\emptyset, \emptyset)$ is a blind pair of equivalent substitutions.*
3. *$\emptyset$ is a blind consistent hedge.*

Proof. We only need to check that all detected process actions preserve the consistency of the environment. This is trivially true if no process actions can be detected by the environment.

1. A frame-theory pair with an empty frame can not detect any process actions. Obviously, $(\emptyset, \emptyset)$ is consistent.
2. There is no expression F such that $\mathrm{n}(F) \subseteq \mathrm{dom}(\emptyset) = \emptyset$. Trivially, $\emptyset \cong \emptyset$.

3. We have that $\mathcal{S}(\emptyset) = \emptyset$, so $\emptyset$ can not detect any process actions. Clearly, $\emptyset$ is a consistent hedge.

□

Note that the strong versions of the bisimilarities do not have any blind environments, since all environments can distinguish betwenn processes that are not $\xrightarrow{\tau}$-bisimilar.

The above propositions imply the following completeness and full abstraction results.

Corollary 2.3.6 *Here we let B be a metavariable for functions that maps all environments to a single blind environment $\top$, appropriate to the codomain of the function. We also let $\sim_\top := \{\top\} \times \mathcal{P} \times \mathcal{P}$, i.e., where $\top \vdash P \sim_\top Q$ for all processes $P, Q \in \mathcal{P}$. Then*

- *The weak versions of $\sim_{\mathrm{a}}$, $\sim_{\mathrm{f}}$, $\sim_{\#}$, $\sim_{\mathrm{h}}$ and $\sim_{\mathrm{s}}$ are B-complete w.r.t. $\sim_{\mathrm{a}}$, $\sim_{\mathrm{f}}$, $\sim_{\#}$, $\sim_{\mathrm{h}}$ and $\sim_{\mathrm{s}}$.*
- *The weak versions of $\sim_{\mathrm{a}}$, $\sim_{\mathrm{f}}$, $\sim_{\#}$, $\sim_{\mathrm{h}}$ and $\sim_{\mathrm{s}}$ are fully B-abstract w.r.t. $\sim_\top$.*

The closest candidate to a dual of blindness turns out to be inconsistency. For the sake of exhibiting this duality, we assume that the inconsistent environments are also members of the environment sets of the bisimilarities.

Proposition 2.3.7 *If there is a $\sim_{\mathrm{y}}$-inconsistent environment c_{y} and $C_{\mathrm{y}} : \mathbf{E}_{\mathrm{x}} \to \mathbf{E}_{\mathrm{y}}$ has* $\mathrm{range}(C_{\mathrm{y}}) = \{c_{\mathrm{y}}\}$ *then $\sim_{\mathrm{y}}$ is C_{y}-sound w.r.t. $\sim_{\mathrm{x}}$.*

Proof. Whenever $C_{\mathrm{y}}(e_{\mathrm{x}}) \vdash P \sim_{\mathrm{y}} Q$ (i.e., never), we have that $e_{\mathrm{x}} \vdash P \sim_{\mathrm{x}} Q$. □

As we have seen, there are inconsistent environments of all types, so if inconsistent environments are assumed to be in the environment domains we get this dual to Corollary 2.3.6.

Corollary 2.3.8 *Here we let C be a metavariable for functions that maps all environments to a single inconsistent environment $\bot$, as appropriate for its codomain. We also let $\sim_\bot := \{\bot\} \times \emptyset \times \emptyset$, i.e., where $\bot \vdash P \not\sim_\bot Q$ for all processes $P, Q \in \mathcal{P}$. Then*

- *$\sim_{\mathrm{a}}$, $\sim_{\mathrm{f}}$, $\sim_{\#}$, $\sim_{\mathrm{h}}$ and $\sim_{\mathrm{s}}$ are C-sound w.r.t. $\sim_{\mathrm{a}}$, $\sim_{\mathrm{f}}$, $\sim_{\#}$, $\sim_{\mathrm{h}}$ and $\sim_{\mathrm{s}}$.*
- *$\sim_{\mathrm{a}}$, $\sim_{\mathrm{f}}$, $\sim_{\#}$, $\sim_{\mathrm{h}}$ and $\sim_{\mathrm{s}}$ are fully C-abstract w.r.t. $\sim_\bot$.*

As we saw above, functions that collapse all environments to a single trivial environment give us the rather non-intuitive result that at least all the weak versions of the bisimilarities defined in Section 2.1 are sound and complete with respect to each other. To avoid this, we seek environment mappings that are sound for blindness (i.e., an environment is blind if its image is blind) and inconsistency.

2.3.3 Full Abstraction and $\mathcal{M}$-equivalence

Another property of all non-trivial environments is synthesis, i.e., the set of message pairs that the environment considers to be equal. As we can see in the definitions of the bisimulations, the synthesis of the environment is fundamental for its interactions with process pairs. On process input, the environment can only generate message pairs in its synthesis (possibly inventing some fresh names), and on process output, the environment cannot accept a message pair which matches one in the synthesis on only one side. We now proceed to prove that non-trivial environments relating the same processes must have the same synthesis.

For the rest of this section, we consider only non-trivial environments, i.e., frame-theory pairs, alleys and hedges and their corresponding bisimilarities of Section 2.1. To compare the synthesis of environments of different types, we extend Definition 2.1.5 as follows.

Definition 2.3.9 *If e_{x} and e_{y} are environments, than we write that $e_{\mathrm{x}} \leq e_{\mathrm{y}}$ if $e_{\mathrm{x}} \vdash M \leftrightarrow N$ implies that $e_{\mathrm{y}} \vdash M \leftrightarrow N$. We say that e_{x} and e_{y} are* $\mathcal{M}$-equivalent*, written $e_{\mathrm{x}} \gtrless e_{\mathrm{y}}$, if $e_{\mathrm{x}} \leq e_{\mathrm{y}}$ and $e_{\mathrm{y}} \leq e_{\mathrm{x}}$.*

As we are mainly interested in full abstractions, it is convenient to have a shorthand for "e_{x} and e_{y} relate the same processes".

Definition 2.3.10 *Two environments e_{x} and e_{y} are* $(\sim_{\mathrm{x}}, \sim_{\mathrm{y}})$-equivalent*, written $e_{\mathrm{x}} \equiv_{\mathrm{y}}^{\mathrm{x}} e_{\mathrm{y}}$ if for all processes P and Q we have that $e_{\mathrm{x}} \vdash P \sim_{\mathrm{x}} Q$ if and only if $e_{\mathrm{y}} \vdash P \sim_{\mathrm{y}} Q$.*

The relation between $\mathcal{M}$-equivalence and $(\sim_{\mathrm{x}}, \sim_{\mathrm{y}})$-equivalence is given by the following proposition. This result implies that in order to be a full abstraction, the environment mapping must be faithful for blindness and consistency, and map a consistent non-blind environment to a $\mathcal{M}$-equivalent counterpart.

Proposition 2.3.11 *If e_{x} and e_{y} are $(\sim_{\mathrm{x}}, \sim_{\mathrm{y}})$-equivalent then either*

- *$e_{\mathrm{x}} \gtrless e_{\mathrm{y}}$ or*
- *e_{x} and e_{y} are both inconsistent or*
- *e_{x} is $\sim_{\mathrm{x}}$-blind and e_{y} is $\sim_{\mathrm{y}}$-blind.*

Proof. If e_{x} is blind, but e_{y} is not, then there exist P, Q such that $e_{\mathrm{y}} \vdash P \sim_{\mathrm{y}} Q$ does not hold. As e_{x} is blind, $e_{\mathrm{x}} \vdash P \sim_{\mathrm{x}} Q$, so $e_{\mathrm{x}} \not\equiv_{\mathrm{y}}^{\mathrm{x}} e_{\mathrm{y}}$. The case where e_{y} is blind, but e_{x} is not, is handled in the same way.

If e_{x} is inconsistent, but e_{y} is not, then $e_{\mathrm{x}} \vdash \mathbf{0} \sim_{\mathrm{x}} \mathbf{0}$ does not hold. As e_{y} is consistent, $e_{\mathrm{y}} \vdash \mathbf{0} \sim_{\mathrm{y}} \mathbf{0}$, so $e_{\mathrm{x}} \not\equiv_{\mathrm{y}}^{\mathrm{x}} e_{\mathrm{y}}$. The case where e_{y} is inconsistent, but e_{x} is not, is handled in the same way.

If e_{x} is neither blind nor inconsistent then there exist names a, b such that $e_{\mathrm{x}} \vdash a \leftrightarrow b$. Let $P_1 = \overline{a}\langle a\rangle.\,\mathbf{0}$ and $Q_1 = (\nu k)\,\overline{b}\langle k\rangle.\,\mathbf{0}$. We have that $e_{\mathrm{x}} \vdash P_1 \not\sim_{\mathrm{x}} Q_1$, since e_{x} can see the difference between a known name and a fresh name (Compare with Proposition 2.2.10 and Proposition 2.2.9, noting the difference between a *known* and a *used* name.). Since $e_{\mathrm{x}} \equiv^{\mathrm{x}}_{\mathrm{y}} e_{\mathrm{y}}$ we get that $e_{\mathrm{y}} \vdash P_1 \not\sim_{\mathrm{y}} Q_1$, which implies that $e_{\mathrm{y}} \vdash a \leftrightarrow n$ or $e_{\mathrm{y}} \vdash n \leftrightarrow b$ for some name n. However, if we let $P_2 = a(z).\,\mathbf{0}$ and $Q_2 = b(z).\,\mathbf{0}$ we have that $e_{\mathrm{x}} \vdash P_2 \sim_{\mathrm{x}} Q_2$. Since $e_{\mathrm{x}} \equiv^{\mathrm{x}}_{\mathrm{y}} e_{\mathrm{y}}$ we get that $e_{\mathrm{y}} \vdash P_2 \sim_{\mathrm{y}} Q_2$, which implies that $e_{\mathrm{y}} \vdash a \leftrightarrow b$.

Now assume that $e_{\mathrm{x}} \vdash M \leftrightarrow N$. Let $P_3 = a(x).[\,x = M\,]\overline{a}\langle a\rangle.\,\mathbf{0}$ and $Q_3 = b(x).\,\mathbf{0}$. We have that $e_{\mathrm{x}} \vdash P_3 \not\sim_{\mathrm{x}} Q_3$, since e_{x} can create and send M. Since $e_{\mathrm{x}} \equiv^{\mathrm{x}}_{\mathrm{y}} e_{\mathrm{y}}$ we get that $e_{\mathrm{y}} \vdash P_3 \not\sim_{\mathrm{y}} Q_3$, which implies that there is e'_{y}, obtained from e_{y} by adding fresh names, such that $e'_{\mathrm{y}} \vdash M \leftrightarrow N'$ for some N'. As $\mathrm{n}(M) \subseteq \mathrm{fn}(P_3)$ we have that no name in $\mathrm{n}(M)$ can be created as fresh, so actually $e_{\mathrm{y}} \vdash M \leftrightarrow N'$.

Let $P_4 = \overline{a}\langle M\rangle.\,\mathbf{0}$ and $Q_4 = \overline{b}\langle N\rangle.\,\mathbf{0}$. Since $e_{\mathrm{x}} \vdash P_4 \sim_{\mathrm{x}} Q_4$ we must have $e_{\mathrm{y}} \vdash P_4 \sim_{\mathrm{y}} Q_4$ which is true only if $e_{\mathrm{y}} \vdash M \leftrightarrow N$, since a consistent environment can not consider M equivalent to two different messages N and N'.

We have now shown that $e_{\mathrm{x}} \leq e_{\mathrm{y}}$. By symmetry, we have that $e_{\mathrm{x}} \geqslant e_{\mathrm{y}}$ when $e_{\mathrm{x}} \equiv^{\mathrm{x}}_{\mathrm{y}} e_{\mathrm{y}}$ and neither e_{x} nor e_{y} is blind nor inconsistent. □

To summarize: An environment mapping should be sound for blindness and inconsistency and preserve the synthesis. We now turn to the task of finding such mappings, and verifying whether they are full abstractions.

2.4 Comparing Environments

Our relations on environment-sensitive bisimulations are based on a comparison of the various environments. In this section, we introduce mappings between frame-theory pairs (**FT**), hedges (**H**) and substitution pairs (**SS**).

Accompanying these mappings, we assemble a "toolbox" of auxiliary results that are used to relate hedges and the other kinds of environments, notably frames and fences.

2.4.1 Properties of Hedges

As we intend to use hedged bisimilarity as the yardstick against which all other bisimilarities are measured, we offer a preliminary investigation of the properties of hedges is offered, that we will exploit in proofs later on.

From the definition of hedges, we immediately get

Lemma 2.4.1 *If* $h \vdash M \leftrightarrow N$ *and* $M \in \mathcal{N}$ *or* $N \in \mathcal{N}$*, then* $(M, N) \in h$*.*

Proof. Clearly SYN-ENC can not derive $h \vdash M \leftrightarrow N$. □

We define a pre-ordering on hedges as follows:

Definition 2.4.2 $g \leq h$ *iff* $\mathcal{S}(g) \subseteq \mathcal{S}(h)$*. If* $g \leq h$ *and* $h \leq g$ *we say that* g *and* h *are* $\mathcal{M}$-equivalent*, written* $g \gtrless h$*.*

$\leq$ is transitive and reflexive by the same properties for $\subseteq$. For results on antisymmetry, see Corollary 2.4.12 and this example:

Example 2.4.3 *Let*

$$g_1 = \{(a, a), (c, k), (\mathrm{E}_k(b), \mathrm{E}_l(a)), (\mathrm{E}_a(c), \mathrm{E}_a(k))\}$$
$$g_2 = \{(a, a), (c, k), (\mathrm{E}_k(b), \mathrm{E}_l(a)), (\mathrm{E}_c(\mathrm{E}_k(b)), \mathrm{E}_k(\mathrm{E}_l(a)))\}$$

Then $g_1 \gtrless g_2$*. More generally, if* h *is a hedge,* $h_1 \leq h$ *and* $h_2 \leq h$ *then* $h \cup h_1 \gtrless h \cup h_2$*.*

An alternative characterization of $\leq$ is as follows:

Lemma 2.4.4 $g \leq h$ *iff* $g \subseteq \mathcal{S}(h)$

Proof. $g \leq h \Rightarrow g \subseteq \mathcal{S}(h)$, since $g \subseteq \mathcal{S}(g) \subseteq \mathcal{S}(h)$.

For the other direction, assume that $g \subseteq \mathcal{S}(h)$ and take any M, N such that $g \vdash M \leftrightarrow N$. By induction on the derivation of $g \vdash M \leftrightarrow N$, we have that $h \vdash M \leftrightarrow N$. □

Corollary 2.4.5 *Some properties relating* $\leq$ *and set operations:*

1. *If* $g \subseteq h$ *then* $g \leq h$*.*
2. *If* $g \leq h$ *and* $f \leq h$ *then* $(f \cup g) \leq h$*.*
3. *If* $g_1 \leq h_1$ *and* $g_2 \leq h_2$ *then* $(g_1 \cup g_2) \leq (h_1 \cup h_2)$*.*

Irreducible Hedges

An interesting subset of **H** is the set of *irreducible* hedges. These are intuitively hedges that are reduced as far as possible, in the sense that no pair of messages in them can be decrypted using the information in the hedge. These results are later used to relate hedges with other types of environments, especially regarding how the environment reacts to process output.

Definition 2.4.6 *A hedge* h *is* irreducible *if* $h = \mathcal{I}(h)$*.* h *is* reducible *if* h *is not irreducible.*

An alternative definition is as follows:

Lemma 2.4.7 *A hedge h is irreducible iff the following condition holds: If $(\mathrm{E}_a(M), \mathrm{E}_b(N)) \in h$ then $(a, b) \notin h$.*

Proof. If this holds then we can not apply HEDGE-DEC to any pair in h, so $\mathcal{A}(h) = h$. By the definition of $\mathcal{I}(h)$ we then have that $\mathcal{I}(h) = h$.

If h is irreducible then the condition holds by the definition of $\mathcal{I}(h)$. □

Corollary 2.4.8 *$\mathcal{I}(h)$ is irreducible for all hedges h.*

Example 2.4.9 *All hedges defined in Example 2.1.17 are irreducible, but neither of the hedges defined in Example 2.4.3. Moreover, if g is reducible and h is any hedge, then $g \cup h$ is reducible.*

As might be expected, the irreducibles of a hedge can be used to generate any message that can be generated by the hedge.

Lemma 2.4.10 *For any hedge h, $h \leq \mathcal{A}(h) \gtrless \mathcal{I}(h)$.*

Proof. As $h \subseteq \mathcal{A}(h) \supseteq \mathcal{I}(h)$ Corollary 2.4.5(1) gives that $h \leq \mathcal{A}(h) \geq \mathcal{I}(h)$. What remains to be proved is $\mathcal{A}(h) \leq \mathcal{I}(h)$. By Lemma 2.4.4, it suffices to show that $\mathcal{A}(h) \subseteq \mathcal{S}(\mathcal{I}(h))$. Assuming that $(M, N) \in \mathcal{A}(h)$, we get $\mathcal{I}(h) \vdash M \leftrightarrow N$ by structural induction on M. □

An irreducible hedge is a subset of any $\mathcal{M}$-equivalent hedge.

Lemma 2.4.11 *If $g \gtrdot h$ and g is irreducible then $g \subseteq h$.*

Proof. Take any $(M, N) \in g$. As $g \gtrdot h$, $h \vdash M \leftrightarrow N$. We have two cases:

If $M \in \mathcal{N}$ or $N \in \mathcal{N}$ then $(M, N) \in h$ by Lemma 2.4.1.

Else, $M = \mathrm{E}_a(M')$ and $N = \mathrm{E}_b(N')$. Since g is irreducible $(a, b) \notin g$ by Lemma 2.4.7. By Lemma 2.4.1 $g \vdash a \not\leftrightarrow b$, so $h \vdash a \not\leftrightarrow b$ and Syn-enc can not derive $h \vdash M \leftrightarrow N$. This shows that $(M, N) \in h$. □

Two $\mathcal{M}$-equivalent irreducible hedges are equal, so the pre-order $\leq$ is an ordering relation on the set of irreducible hedges.

Corollary 2.4.12 *If $g \gtrdot h$ and both g and h are irreducible, then $g = h$.*

The ordering of hedges is preserved by $\mathcal{I}(\cdot)$.

Lemma 2.4.13 *If $g \leq h$, then $\mathcal{I}(g) \leq \mathcal{I}(h)$.*

Proof. According to Lemma 2.4.10 $\mathcal{I}(g) \leq \mathcal{A}(g)$, so by the transitivity of $\leq$ we only need to show $\mathcal{A}(g) \leq \mathcal{I}(h)$. By Lemma 2.4.4 this holds iff $\mathcal{A}(g) \subseteq \mathcal{S}(\mathcal{I}(h))$, which we show by induction on the derivation of $\mathcal{A}(g)$. Take any $(M, N) \in \mathcal{A}(g)$.

The base case is that $(M, N) \in g$. By Lemma 2.4.10 $h \leq \mathcal{I}(h)$, so $g \leq \mathcal{I}(h)$ by the transitivity of $\leq$. In particular, $\mathcal{I}(h) \vdash M \leftrightarrow N$.

Otherwise we used Ana-dec (Definition 2.1.18) to derive $(M, N) \in \mathcal{A}(g)$, so there are a and b such that $(\mathrm{E}_a(M), \mathrm{E}_b(N)) \in \mathcal{A}(g)$ and $(a, b) \in \mathcal{A}(g)$. By induction $\mathcal{I}(h) \vdash \mathrm{E}_a(M) \leftrightarrow \mathrm{E}_b(N)$ and $\mathcal{I}(h) \vdash a \leftrightarrow b$. By Lemma 2.4.1 $(a, b) \in \mathcal{I}(h)$, so $(\mathrm{E}_a(M), \mathrm{E}_b(N)) \notin \mathcal{I}(h)$ by the definition of $\mathcal{I}(\cdot)$. Then Syn-enc must have been used to derive $\mathcal{I}(h) \vdash \mathrm{E}_a(M) \leftrightarrow \mathrm{E}_b(N)$, which gives $\mathcal{I}(h) \vdash M \leftrightarrow N$. □

The irreducibles of two $\mathcal{M}$-equivalent hedges are equal.

Lemma 2.4.14 *If $g \gtrdot h$, then $\mathcal{I}(g) = \mathcal{I}(h)$.*

Proof. $\mathcal{I}(g) \gtrdot \mathcal{I}(h)$ by Lemma 2.4.13. $\mathcal{I}(g)$ and $\mathcal{I}(h)$ are both irreducible by Corollary 2.4.8. The equality then follows from Corollary 2.4.12. □

We also have this more general variant of Corollary 2.4.8, that is used to prove that hedges and alleys behave similarly on process output.

Lemma 2.4.15 *If g and h are hedges, then $\mathcal{I}(\mathcal{I}(h) \cup g) = \mathcal{I}(h \cup g)$.*

Proof. By Corollary 2.4.5(1) $g \leq h \cup g$, so $\mathcal{I}(g) \leq \mathcal{I}(h \cup g)$ by Lemma 2.4.13. By Lemma 2.4.10 $g \leq \mathcal{I}(g)$, so by transitivity $g \leq \mathcal{I}(h \cup g)$. For $\mathcal{I}(h)$ we know that $\mathcal{I}(h) \subseteq \mathcal{A}(h) \subseteq \mathcal{A}(h \cup g) \leq \mathcal{I}(h \cup g)$, where the last relation is due to Lemma 2.4.10. By Corollary 2.4.5(2) $\mathcal{I}(h) \cup g \leq \mathcal{I}(h \cup g)$, so $\mathcal{I}(\mathcal{I}(h) \cup g) \leq \mathcal{I}(\mathcal{I}(h \cup g))$ by Lemma 2.4.13. $\mathcal{I}(h \cup g)$ is irreducible by Corollary 2.4.8, so $\mathcal{I}(\mathcal{I}(h) \cup g) \leq \mathcal{I}(h \cup g)$.

$h \leq \mathcal{I}(h)$ by Lemma 2.4.10, so by Corollary 2.4.5(3) we have that $h \cup g \leq \mathcal{I}(h) \cup g$. By Lemma 2.4.13 we have that $\mathcal{I}(h \cup g) \leq \mathcal{I}(\mathcal{I}(h) \cup g)$, so $\mathcal{I}(h \cup g) \gtrsim \mathcal{I}(\mathcal{I}(h) \cup g)$. The equality now follows from Corollary 2.4.12. □

Proper Care for Consistent Hedges

As we have seen in Section 2.3, an environment mapping should preserve consistency. In order to show that this holds for the environment mappings we will define, we investigate consistent hedges and their properties.

Lemma 2.4.16 *If h is consistent then h is irreducible.*

Proof. By condition 3 for consistency, we have that $(\mathrm{E}_a(M), \mathrm{E}_b(N)) \in h$ implies $(a, b) \notin h$. By Lemma 2.4.7 this means that h is irreducible. □

Note that we have only used a special case of condition 3 in the proof of Lemma 2.4.16. The three conditions for consistency are pairwise disjoint (see Example 2.1.17), so consistency is a much stronger constraint then irreducibility.

A generalized version of condition 2 for consistency is that a consistent hedge can not generate two message pairs that differ in only one component.

Lemma 2.4.17 *Let h be consistent with $h \vdash M \leftrightarrow N$ and $h \vdash M' \leftrightarrow N'$. Then $M = M'$ iff $N = N'$.*

Proof. By symmetry we need only study the case $M = M'$. The proof is by induction on the derivation of $h \vdash M \leftrightarrow N$.

If $(M, N) \in h$ we will first show that $(M, N') \in h$. If M is a name this follows from Lemma 2.4.1. Otherwise $M = \mathrm{E}_a(K)$, but since $a \notin \pi_1(h)$ by condition 3 of consistency we can not use Syn-enc to derive $h \vdash M \leftrightarrow N'$. Now we know that $(M, N) \in h$ and $(M, N') \in h$, so $N = N'$ by condition 2 for consistency.

If $M = \mathrm{E}_a(K)$, $N = \mathrm{E}_b(L)$, $h \vdash K \leftrightarrow L$ and $h \vdash a \leftrightarrow b$ then $a \in \pi_1(h)$ by Lemma 2.4.1. As h is consistent, $M \notin \pi_1(h)$, so we must have used Syn-enc to derive $h \vdash M \leftrightarrow N'$. This gives that $N' = \mathrm{E}_c(L')$ for some c, L' such that $h \vdash K \leftrightarrow L'$ and $h \vdash a \leftrightarrow c$. By induction $L = L'$ and $b = c$. □

Two $\mathcal{M}$-equivalent consistent hedges are always equal.

Lemma 2.4.18 *If $g \gtrless h$ and both g and h are consistent, then $g = h$.*

Proof. g and h are irreducible by Lemma 2.4.16. The equality follows from Corollary 2.4.12. □

Any irreducible "trimming" of a consistent hedge is consistent.

Lemma 2.4.19 *If h is consistent, g is irreducible and $g \leq h$ then g is consistent.*

Proof. Assume that $(M, N) \in g$ and note that $h \vdash M \leftrightarrow N$. We only need to show one direction of the symmetric conditions.

1. If $M \in \mathcal{N}$ then $(M, N) \in h$ by Lemma 2.4.1, so $N \in \mathcal{N}$ as h is consistent.
2. See Lemma 2.4.17.
3. Assume that $M = \mathrm{E}_a(K)$. If $(M, N) \in h$ then $a \notin \pi_1(h)$ by condition 3 for consistency, so $a \notin \pi_1(g)$ by Lemma 2.4.1.
   Else Syn-enc has been used to derive $h \vdash M \leftrightarrow N$, so $N = \mathrm{E}_b(L)$ where $h \vdash a \leftrightarrow b$ and $h \vdash K \leftrightarrow L$. We then show that there is no N' such that $(a, N') \in g$ by contradiction. For $N' = b$, $(a, b) \in g$ would contradict that g is irreducible by Lemma 2.4.7. For any $N' \neq b$ we have by Lemma 2.4.17 that $h \vdash a \not\leftrightarrow N'$, so $(a, N') \notin g$.

□

Disjoint consistent hedges may be directly combined.

Lemma 2.4.20 *If g and h are consistent and $\mathrm{n}(g) \cap \mathrm{n}(h) = \emptyset$, then $g \cup h$ is consistent.*

Proof. Take any $(M, N) \in g \cup h$. By symmetry we may assume that $(M, N) \in g$.

1. $M \in \mathcal{N} \iff N \in \mathcal{N}$ is clear, since g is consistent.
2. Take any $(M', N') \in g \cup h$. As $\mathrm{n}(M) \neq \emptyset \neq \mathrm{n}(N)$ we have that $(M', N') \in g$ whenever $M = M'$ or $N = N'$. As g is consistent, $M = M'$ iff $N = N'$.
3. If $M = \mathrm{E}_a(M')$ and $N = \mathrm{E}_b(N')$ then $a \notin \pi_1(g)$ and $b \notin \pi_2(g)$ as g is consistent. As $\{a, b\} \subseteq \mathrm{n}(g)$ we have that $\{a, b\} \cap \mathrm{n}(h) = \emptyset$ and as a special case of this, $a \notin \pi_1(h)$ and $b \notin \pi_2(h)$.

□

2.4.2 Frames and Hedges

Since the definitions of hedges and frame-theory pairs are similar, the correspondence is fairly clear. However, as pointed out in Section 2.2, hedged and framed bisimilarity do not coincide. In Section 2.5, we will prove that framed bisimularity implies hedged bisimularity. We begin with some results relating frame-theory pairs and hedges.

Definition 2.4.21 *The hedge corresponding to a frame-theory pair is defined by:*

$$\psi : \mathbf{FT} \to \mathbf{H} : \psi(\mathrm{fr}, \mathrm{th}) := \mathrm{Id}_{\mathrm{fr}} \cup \mathrm{th}$$

Note that $\mathcal{S}(\mathrm{fr}, \mathrm{th}) = \mathcal{S}(\psi(\mathrm{fr}, \mathrm{th}))$, so $(\mathrm{fr}, \mathrm{th}) \leq (\mathrm{fr}', \mathrm{th}')$ iff $\psi(\mathrm{fr}, \mathrm{th}) \leq \psi(\mathrm{fr}', \mathrm{th}')$. We thus have that ψ preserves the synthesis of frame-theory pairs, which was one of the desirable properties mentioned in Section 2.3.

Since consistent frame-theory pairs have names only in the frame, we have this obvious strengthening of Lemma 2.4.1.

Lemma 2.4.22 *If* $(\mathrm{fr}, \mathrm{th})$ *is consistent and* $(\mathrm{fr}, \mathrm{th}) \vdash M \leftrightarrow N$ *where* $M \in \mathcal{N}$ *or* $N \in \mathcal{N}$ *then* $M = N \in \mathrm{fr}$.

Proof. Since Syn-enc can not derive $(\mathrm{fr}, \mathrm{th}) \vdash M \leftrightarrow N$ we have that $(M, N) \in \psi(\mathrm{fr}, \mathrm{th})$. There are no names in the theory by condition 1 for the consistency of $(\mathrm{fr}, \mathrm{th})$, so $M = N \in \mathrm{fr}$. □

An important result is that if a frame-theory pair is consistent then its corresponding hedge is also consistent. In other words, ψ preserves consistency.

Lemma 2.4.23 *If* $(\mathrm{fr}, \mathrm{th})$ *is consistent then* $\psi(\mathrm{fr}, \mathrm{th})$ *is consistent.*

Proof. Take any $(M, N) \in \psi(\mathrm{fr}, \mathrm{th})$. We show only one direction of the symmetric conditions.

1. $M \in \mathcal{N} \iff N \in \mathcal{N}$ by Lemma 2.4.22.
2. Assume that $(M, N') \in \psi(\mathrm{fr}, \mathrm{th})$. If $\{M, N, N'\} \cap \mathcal{N} \neq \emptyset$ then $M = N = N' \in \mathrm{fr}$ by Lemma 2.4.22. Else, $(M, N) \in \mathrm{th}$ and $(M, N') \in \mathrm{th}$, so $N = N'$ by condition 2 of the consistency of $(\mathrm{fr}, \mathrm{th})$.
3. If $M = \mathrm{E}_a(M')$ then according to Lemma 2.4.22 $a \in \pi_1(\psi(\mathrm{fr}, \mathrm{th}))$ only if $a \in \mathrm{fr}$, which is false by condition 3 for the consistency of $(\mathrm{fr}, \mathrm{th})$.

□

Note that the reverse implication does not hold. For example, $(\emptyset, \{(a, b)\})$ is not a consistent frame-theory pair, but $\psi(\emptyset, \{(a, b)\}) = \{(a, b)\}$ is a consistent hedge. We will also need an extension of Lemma 2.4.19 to consistent frame-theory pairs.

Lemma 2.4.24 *If* $(\mathrm{fr}, \mathrm{th})$ *is consistent,* h *is irreducible and* $h \leq \psi(\mathrm{fr}, \mathrm{th})$ *there exist* $\mathrm{fr}', \mathrm{th}'$ *such that* $h = \psi(\mathrm{fr}', \mathrm{th}')$ *and* $(\mathrm{fr}', \mathrm{th}')$ *is consistent.*

Proof. We show that $\mathrm{fr}' = \pi_1(h \cap (\mathcal{N} \times \mathcal{N})), \mathrm{th}' = h \setminus \mathrm{Id}_{\mathrm{fr}'}$ have the desired properties. Note that $h \subseteq \mathcal{S}(\psi(\mathrm{fr}, \mathrm{th}))$ by Lemma 2.4.4.

1. We first show that $\mathrm{Id}_{\mathrm{fr}'} = h \cap (\mathcal{N} \times \mathcal{M}) = h \cap (\mathcal{M} \times \mathcal{N})$. Take $(M, N) \in h$ such that $M \in \mathcal{N}$ or $N \in \mathcal{N}$. By Lemma 2.4.22 applied to $\psi(\mathrm{fr}, \mathrm{th}) \vdash M \leftrightarrow N$ we have that $M = N \in \mathrm{fr}$, so $M = N \in \mathrm{fr}'$ by definition. On the other hand, whenever $a \in \mathrm{fr}'$ then there is by definition b such that $(a, b) \in h$. As above $a = b$ by Lemma 2.4.22.
2. We then show that $\psi(\mathrm{fr}', \mathrm{th}') = h$. By the definition of $(\mathrm{fr}', \mathrm{th}')$ we have that $\psi(\mathrm{fr}', \mathrm{th}') = \mathrm{th}' \cup \mathrm{Id}_{\mathrm{fr}'} = (h \setminus \mathrm{Id}_{\mathrm{fr}'}) \cup \mathrm{Id}_{\mathrm{fr}'} = h \cup \mathrm{Id}_{\mathrm{fr}'} \supseteq h$. We have equality if $\mathrm{Id}_{\mathrm{fr}'} \subseteq h$, which follows from step 1.
3. Now we show the consistency of $(\mathrm{fr}', \mathrm{th}')$. First $\psi(\mathrm{fr}, \mathrm{th})$ is consistent by Lemma 2.4.23, so h is consistent by Lemma 2.4.19. Take any $(M, N) \in \mathrm{th}'$ and note that $\psi(\mathrm{fr}, \mathrm{th}) \vdash M \leftrightarrow N$.
   (a) If $M \in \mathcal{N}$ or $N \in \mathcal{N}$ then by step 1 we have that $(M, N) \notin h \setminus \mathrm{Id}_{\mathrm{fr}'} = \mathrm{th}'$.
   (b) If $(M', N') \in \mathrm{th}'$ then we have that $\psi(\mathrm{fr}, \mathrm{th}) \vdash M' \leftrightarrow N'$, so $M = M' \iff N = N'$ by Lemma 2.4.17.
   (c) Assume that $M = \mathrm{E}_a(M')$ and $N = \mathrm{E}_b(N')$. By the consistency of h we have that $a \notin \pi_1(h)$ and $b \notin \pi_2(h)$, so by step 1 we have that $a, b \notin \mathrm{fr}'$.

□

2.4.3 Fences and Hedges

Informally, ξ (see Table 2.1 on page 23) is a function that extends a given consistent frame-theory pair with a new pair of messages. If the resulting frame-theory pair is not consistent, ξ returns $\perp$. Since the results of [EHHO99] hold for a superset of $\mathcal{M}$ it is easy to see that they hold for the message syntax of this paper. To show the soundness of fenced bisimulation with respect to hedged we exhibit that ξ is a sound approximation of $\mathcal{I}(\cdot)$. We start with two results proved by [EHHO99].

Soundness: ξ always creates a consistent extension.

Lemma 2.4.25 *If* $(\mathrm{fr}, \mathrm{th})$ *is consistent and* $\xi(\mathrm{fr}, \mathrm{th}, M, N) \neq \perp$ *then*

1. $\xi(\mathrm{fr}, \mathrm{th}, M, N)$ *is consistent*
2. $\xi(\mathrm{fr}, \mathrm{th}, M, N) \vdash M \leftrightarrow N$
3. $(\mathrm{fr}, \mathrm{th}) \leq \xi(\mathrm{fr}, \mathrm{th}, M, N)$

Completeness: If there exists a consistent extension then ξ is defined and returns the smallest of all consistent extensions.

Lemma 2.4.26 *If* $(\mathrm{fr}, \mathrm{th})$ *and* $(\mathrm{fr}', \mathrm{th}')$ *are consistent,* $(\mathrm{fr}, \mathrm{th}) \leq (\mathrm{fr}', \mathrm{th}')$ *and* $(\mathrm{fr}', \mathrm{th}') \vdash M \leftrightarrow N$ *then* $\xi(\mathrm{fr}, \mathrm{th}, M, N) \neq \bot$ *and* $\xi(\mathrm{fr}, \mathrm{th}, M, N) \leq (\mathrm{fr}', \mathrm{th}')$.

Now we can show the relationship between ξ and $\mathcal{I}(\cdot)$, namely that whenever a consistent fence accepts a message pair, its corresponding hedge also does. That the converse does not hold was shown in Section 2.2.

Lemma 2.4.27 *If* $(\mathrm{fr}, \mathrm{th})$ *is consistent and* $\xi(\mathrm{fr}, \mathrm{th}, M, N) \neq \bot$ *then*

$$\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\}) = \psi(\xi(\mathrm{fr}, \mathrm{th}, M, N))$$

Proof. Let $g = \psi(\mathrm{fr}, \mathrm{th})$, $g' = g \cup \{(M, N)\}$ and $h_\xi = \psi(\xi(\mathrm{fr}, \mathrm{th}, M, N))$. We first show that $\mathcal{I}(g') \leq h_\xi$, then the other direction.

By Lemma 2.4.25 $\xi(\mathrm{fr}, \mathrm{th}, M, N) \vdash M \leftrightarrow N$ and $(\mathrm{fr}, \mathrm{th}) \leq \xi(\mathrm{fr}, \mathrm{th}, M, N)$. After application of ψ this gives that $h_\xi \vdash M \leftrightarrow N$ and $g \leq h_\xi$, so the combined hedge $g' \leq h_\xi$ by Corollary 2.4.5(3). Reducing both sides, Lemma 2.4.13 gives $\mathcal{I}(g') \leq \mathcal{I}(h_\xi)$. By Lemma 2.4.25 we have that $\xi(\mathrm{fr}, \mathrm{th}, M, N)$ is consistent, so h_ξ is consistent by Lemma 2.4.23. h_ξ is then irreducible by Lemma 2.4.16, so $\mathcal{I}(g') \leq h_\xi$.

Since $\mathcal{I}(g')$ is irreducible by Corollary 2.4.8 we can apply Lemma 2.4.24 to $\mathcal{I}(g') \leq h_\xi$, giving us a consistent $(\mathrm{fr}', \mathrm{th}')$ such that $\mathcal{I}(g') = \psi(\mathrm{fr}', \mathrm{th}')$. We have by Lemma 2.4.10 that $g' \leq \mathcal{I}(g')$, which can be divided into $\mathcal{I}(g') \vdash M \leftrightarrow N$ and $g \leq \mathcal{I}(g')$. In the framed world, this means that $(\mathrm{fr}', \mathrm{th}') \vdash M \leftrightarrow N$ and $(\mathrm{fr}, \mathrm{th}) \leq (\mathrm{fr}', \mathrm{th}')$. Lemma 2.4.26 then yields $\xi(\mathrm{fr}, \mathrm{th}, M, N) \leq (\mathrm{fr}', \mathrm{th}')$. Going back to the hedges, this means that $h_\xi \leq \mathcal{I}(g')$.

We have now proved the inequality in both directions, so $h_\xi \geqslant \mathcal{I}(g')$ and the equality follows from Corollary 2.4.12. □

2.4.4 Hedges and Alleys

In this section we show that hedges are in a certain sense equivalent to the environments of alley bisimulation. As the theories are fairly different, we need a number of auxiliary results to tie the two representations together.

Lemmas on Substitutions

We first recall some lemmas proved by [BDP02]:

Lemma 2.4.28 $\forall \sigma, x \in \mathrm{dom}(\sigma) : \exists G : \mathrm{n}(G) \subseteq \mathrm{dom}(\sigma) \wedge \mathbf{e}(G\sigma) = \mathcal{C}(\sigma, x)$ *(cf. Definition 2.1.9).*

Lemma 2.4.29 *For* $\sigma \cong \rho$ *where* $\mathrm{dom}(\sigma) = \{x_i\}_{i \in I}$ *we let* $M_i = \mathcal{C}(\sigma, x_i)$ *and* $N_i = \mathcal{C}(\rho, x_i)$. *Then, for every* G *such that* $\mathrm{n}(G) \subseteq \mathrm{dom}(\sigma)$ *either:*

1. $\mathbf{e}(G\sigma) = \mathbf{e}(G\rho) = \bot$, *or*
2. *There are* i *and a tuple* $\tilde{\iota}$ *of indices from* I *such that*

$$\begin{aligned} \mathbf{e}(G\sigma) &= \mathrm{E}_{M_{\iota_m}}(\cdots \mathrm{E}_{M_{\iota_2}}(\mathrm{E}_{M_{\iota_1}}(M_i))\cdots) \\ \mathbf{e}(G\rho) &= \mathrm{E}_{N_{\iota_m}}(\cdots \mathrm{E}_{N_{\iota_2}}(\mathrm{E}_{N_{\iota_1}}(N_i))\cdots)\,. \end{aligned}$$

As an abbreviation, whenever $\mathrm{dom}(\sigma) = \{x_i\}_{i\in I}$ and $\{\tilde{\iota}\} \subseteq I$ we write $\mathrm{E}_{\tilde{\iota}}(\sigma_i)$ for $\mathrm{E}_{\mathcal{C}(\sigma,x_{\iota_m})}(\cdots \mathrm{E}_{\mathcal{C}(\sigma,x_{\iota_2})}(\mathrm{E}_{\mathcal{C}(\sigma,x_{\iota_1})}(\mathcal{C}(\sigma,x_i)))\cdots)$

Lemma 2.4.30 *If* $\sigma \cong \rho$ *and* $\mathrm{n}(G) \subseteq \mathrm{dom}(\sigma)$ *with* $\mathbf{e}(G\sigma) \neq \bot$ *then* $\sigma\{^{\mathbf{e}(G\sigma)}/_x\} \cong \rho\{^{\mathbf{e}(G\rho)}/_x\}$.

Lemma 2.4.31 *If* $\sigma\{^{M}/_x\} \cong \rho\{^{N}/_x\}$ *then* $\sigma \cong \rho$.

Lemma 2.4.32 *If* $M \in \mathcal{A}(\sigma)$ *then there are* i *and* $\tilde{\iota}$ *such that* $M = \mathrm{E}_{\tilde{\iota}}(\sigma_i)$.

Lemma 2.4.33 $\mathrm{fn}(\sigma) = \mathrm{n}(\mathcal{I}(\sigma))$

From Alleys to Hedges

We first need some natural way to move back and forth between consistent hedges and consistent pairs of substitutions. Recall that **SS** is the set of pairs of substitutions with the same domain. Given such a substitution pair, we can use the cores to get a hedge as follows:

Definition 2.4.34 *The hedge corresponding to a substitution pair is defined by:*

$$\varphi : \mathbf{SS} \to \mathbf{H} : \varphi(\sigma,\rho) := \{\, (\mathcal{C}(\sigma,x), \mathcal{C}(\rho,x)) \mid x \in \mathrm{dom}(\sigma) \,\}$$

Lemma 2.4.35 *If* $\sigma \cong \rho$ *then* $\varphi(\sigma,\rho)$ *is consistent.*

Proof. Take any $(M,N) \in \varphi(\sigma,\rho)$ and $x \in \mathrm{dom}(\sigma)$ such that $M = \mathcal{C}(\sigma,x)$ and $N = \mathcal{C}(\rho,x)$.

1. By condition 1 for $\sigma \cong \rho$ we have that $M \in \mathcal{N}$ iff $N \in \mathcal{N}$.
2. Take any $(M',N') \in \varphi(\sigma,\rho)$ and $x' \in \mathrm{dom}(\sigma)$ such that $M' = \mathcal{C}(\sigma,x')$ and $N' = \mathcal{C}(\rho,x')$. By condition 2 for $\sigma \cong \rho$ we have that $M = M'$ iff $N = N'$.
3. Since $M = \mathcal{C}(\sigma,x) \in \mathcal{I}(\sigma)$ it can not be decrypted by any key $a \in \mathcal{A}(\sigma)$. By the definition of $\mathcal{I}(\cdot)$, $a \in \mathcal{A}(\sigma)$ iff $a \in \mathcal{I}(\sigma) = \{\mathcal{C}(\sigma,x) | x \in \mathrm{dom}(\sigma)\} = \pi_1(\varphi(\sigma,\rho))$. A symmetrical argument holds for $\mathcal{C}(\rho,x_i)$.

□

Corollary 2.4.36 $\mathrm{fn}(\sigma) = \mathrm{n}(\pi_1(\varphi(\sigma,\rho)))$ *and* $\mathrm{fn}(\rho) = \mathrm{n}(\pi_2(\varphi(\sigma,\rho)))$

The following lemma gives an alternative definition of $\varphi(\sigma,\rho)$, showing that $\mathcal{I}(h)$ computes exactly matching pairs of cores.

Lemma 2.4.37 *Let* $\sigma \cong \rho$ *and* $h = \{\, (\sigma(x_i), \rho(x_i)) \mid x_i \in \mathrm{dom}(\sigma) \,\}$.
Then $\varphi(\sigma,\rho) = \mathcal{I}(h)$.

Proof.

1. First we show that whenever $M \in \mathcal{A}(\sigma)$ there are $i, \tilde{\imath}$ such that $M = \mathrm{E}_{\tilde{\imath}}(\sigma_i)$, $\mathrm{E}_{\tilde{\imath}}(\rho_i) \in \mathcal{A}(\rho)$ and $(M, \mathrm{E}_{\tilde{\imath}}(\rho_i)) \in \mathcal{A}(h)$. We use induction on the derivation of $M \in \mathcal{A}(\sigma)$.
   If $M = \sigma(x_i)$ then we get $i, \tilde{\imath}$ such that $M = \mathrm{E}_{\tilde{\imath}}(\sigma_i)$ and $\mathrm{E}_{\tilde{\imath}}(\rho_i) = \rho(x_i)$ by condition 3 of $\sigma \cong \rho$. By definition, $(M, \mathrm{E}_{\tilde{\imath}}(\rho_i)) \in h \subseteq \mathcal{A}(h)$
   Else, there is a such that $\mathrm{E}_a(M), a \in \mathcal{A}(\sigma)$. By the induction assumption there are $i, \tilde{\imath}$ such that $\mathrm{E}_a(M) = \mathrm{E}_{\tilde{\imath}}(\sigma_i)$, $N = \mathrm{E}_{\tilde{\imath}}(\rho_i)$, $(\mathrm{E}_a(M), N) \in \mathcal{A}(h)$ and $(a, \mathcal{C}(\rho, x_{\iota_m})) \in \mathcal{A}(h)$ where $m = |\tilde{\imath}|$. If $\tilde{\imath}' = \iota_1\iota_2\ldots\iota_{m-1}$ then $M = \mathrm{E}_{\tilde{\imath}'}(\sigma_i)$ and using HEDGE-DEC we get that $(M, \mathrm{E}_{\tilde{\imath}'}(\rho_i)) \in \mathcal{A}(h)$. As $\mathcal{C}(\rho, x_{\iota_m}) \in \mathcal{A}(\rho)$ we have $\mathrm{E}_{\tilde{\imath}'}(\rho_i) \in \mathcal{A}(\rho)$ by SET-DEC.
2. We show that if $(M, N) \in \mathcal{A}(h)$ then there are $i, \tilde{\imath}$ with $M = \mathrm{E}_{\tilde{\imath}}(\sigma_i) \in \mathcal{A}(\sigma)$ and $N = \mathrm{E}_{\tilde{\imath}}(\rho_i) \in \mathcal{A}(\rho)$ by induction on the derivation of $(M, N) \in \mathcal{A}(h)$. The base case follows from condition 3 of $\sigma \cong \rho$.
   Otherwise, there are a, b such that $(a, b) \in \mathcal{A}(h)$ and $(\mathrm{E}_a(M), \mathrm{E}_b(N)) \in \mathcal{A}(h)$. By the induction assumption there are $j, i, \tilde{\imath}$ with $a = \mathcal{C}(\sigma, x_j) \in \mathcal{A}(\sigma)$, $b = \mathcal{C}(\rho, x_j) \in \mathcal{A}(\rho)$, $\mathrm{E}_a(M) = \mathrm{E}_{\tilde{\imath}}(\sigma_i) \in \mathcal{A}(\sigma)$ and $\mathrm{E}_b(N) = \mathrm{E}_{\tilde{\imath}}(\rho_i) \in \mathcal{A}(\rho)$. By SET-DEC $M \in \mathcal{A}(\sigma)$ and $N \in \mathcal{A}(\rho)$. Clearly $M = \mathrm{E}_{\tilde{\imath}'}(\sigma_i)$ and $N = \mathrm{E}_{\tilde{\imath}'}(\rho_i)$, where $\tilde{\imath}' = \iota_1\iota_2\ldots\iota_{m-1}, m = |\tilde{\imath}|$.

By 2, if $(M, N) \in \mathcal{A}(h)$ then there are $i, \tilde{\imath}$ such that $M = \mathrm{E}_{\tilde{\imath}}(\sigma_i)$ and $N = \mathrm{E}_{\tilde{\imath}}(\rho_i)$. If $|\tilde{\imath}| = m > 0$, we have by 1 that $(\mathcal{C}(\sigma, x_{\iota_m}), \mathcal{C}(\rho, x_{\iota_m})) \in \mathcal{A}(h)$, so $(M, N) \notin \mathcal{I}(h)$. Using 1 we also have that $\forall x_i \in \mathrm{dom}(\sigma) : (\mathcal{C}(\sigma, x_i), \mathcal{C}(\rho, x_i)) \in \mathcal{I}(h)$. □

From Hedges to Alleys

For the transformation of hedges into substitution pairs, we have to invent the domain for the substitutions. However, it does not matter which particular domain we select, since the domain of the substitutions is not important for the definition of alley bisimulation. Note that $\mathcal{M} \times \mathcal{M}$ and $\mathcal{V}$ are both countably infinite, so there is a bijection between them. To invent the substitution domain, we may use any such bijection.

Definition 2.4.38 *Fix a bijection $f : \mathcal{M} \times \mathcal{M} \to \mathcal{V}$. The alley corresponding to a hedge is defined by:*

$$\begin{array}{rrcl} \theta^f : \mathbf{H} \to \mathbf{SS} : & \theta^f(h) & := & (\ \theta_1^f(h)\ ,\ \theta_2^f(h)\ ) \\ & \theta_1^f(h) & := & \{\ \{{}^{M}/_{f(M,N)}\} \mid (M,N) \in h\ \} \\ & \theta_1^f(h) & := & \{\ \{{}^{N}/_{f(M,N)}\} \mid (M,N) \in h\ \} \end{array}$$

We sometimes use the projections $h_1^f := \theta_1^f(h)$ and $h_2^f := \theta_2^f(h)$ to denote the left and right substitutions corresponding to h (under f).

In the following, we implicitly assume a suitable fixed bijection f.

Lemma 2.4.39 *If h is consistent then $h_1^f \cong h_2^f$ and $h = \varphi(\theta^f(h))$.*

Proof. Clearly h_1^f and h_2^f have the same domain. By condition 3 for the consistency of h we have that $h_1^f(x) = \mathcal{C}(h_1^f, x)$ and $h_2^f(x_i) = \mathcal{C}(h_2^f, x_i)$, so
$h = \{\ (\mathcal{C}(h_1^f, x_i), \mathcal{C}(h_2^f, x_i)) \mid x_i \in \mathrm{dom}(h_1^f)\ \}$ as desired.

To show that $h_1^f \cong h_2^f$ we fix $x \in \mathrm{dom}(h_1^f)$.

1. $\mathcal{C}(h_1^f, x) \in \mathcal{N}$ iff $\mathcal{C}(h_2^f, x) \in \mathcal{N}$ by condition 1 of the consistency of h.
2. If $y \in \mathrm{dom}(h_1^f)$ then $\mathcal{C}(h_1^f, x) = \mathcal{C}(h_1^f, y)$ and $\mathcal{C}(h_2^f, x) = \mathcal{C}(h_2^f, y)$ iff $x = y$ by condition 2 of the consistency of h.
3. As above, $h_1^f(x) = \mathcal{C}(h_1^f, x)$ and $h_2^f(x) = \mathcal{C}(h_2^f, x)$.

□

Thus, θ^f preserves consistency. Also, since φ is an inverse of θ^f we get that if φ preserves the synthesis, then so does θ^f. We now proceed to show that it is indeed true that φ preserves environment synthesis.

Sending and Receiving Messages

In this paragraph we always implicitly assume that $\sigma \cong \rho$ and that $h = \varphi(\sigma, \rho)$.

The following two lemmas show that the hedge derived from a consistent pair of substitutions can generate exactly the same message pairs as the substitutions. In other words, φ preserves the synthesis of environments.

Lemma 2.4.40 *If $(\sigma, \rho) \vdash M \leftrightarrow N$ then $h \vdash M \leftrightarrow N$.*

Proof. By the definition of $(\sigma, \rho) \vdash M \leftrightarrow N$ we get that there exits G such that $\mathrm{n}(G) \subseteq \mathrm{dom}(\sigma)$, $M = \mathbf{e}(G\sigma)$ and $N = \mathbf{e}(G\rho)$.

By Lemma 2.4.29 we get that $(\mathbf{e}(G\sigma), \mathbf{e}(G\rho)) = (\mathrm{E}_{\tilde{\imath}}(\sigma_i), \mathrm{E}_{\tilde{\imath}}(\rho_i))$ for some $i, \tilde{\imath}$. We prove that $h \vdash \mathbf{e}(G\sigma) \leftrightarrow \mathbf{e}(G\rho)$ by induction on $|\tilde{\imath}|$.

- If $|\tilde{\iota}| = 0$, then $(\mathbf{e}(G\sigma), \mathbf{e}(G\rho)) = (\mathcal{C}(\sigma, x_i), \mathcal{C}(\rho, x_i)) \in h$.
- Else, if $|\tilde{\iota}| = m + 1$ and $\tilde{\iota}' = \iota_1\iota_2 \ldots \iota_m$ then $h \vdash \mathrm{E}_{\tilde{\iota}'}(\sigma_i) \leftrightarrow \mathrm{E}_{\tilde{\iota}'}(\rho_i)$ by the induction assumption. As $(\mathcal{C}(\sigma, x_{\iota_{m+1}}), \mathcal{C}(\rho, x_{\iota_{m+1}})) \in h$ we can use Eq-enc to derive that $h \vdash \mathrm{E}_{\tilde{\iota}}(\sigma_i) \leftrightarrow \mathrm{E}_{\tilde{\iota}}(\rho_i)$.

□

Lemma 2.4.41 *If* $h \vdash M \leftrightarrow N$ *then* $(\sigma, \rho) \vdash M \leftrightarrow N$.

Proof. The proof is by induction on the derivation of $h \vdash M \leftrightarrow N$. First note that for each $x_i \in \mathrm{dom}(\sigma)$ we can by Lemma 2.4.28 find an F_i such that $\mathrm{n}(F_i) \subseteq \mathrm{dom}(\sigma)$ and $\mathbf{e}(F_i\sigma) = \mathcal{C}(\sigma, x_i)$. By Lemma 2.4.29, $(\mathbf{e}(F_i\sigma), \mathbf{e}(F_i\rho)) = (\mathcal{C}(\sigma, x_i), \mathcal{C}(\rho, x_i))$.

- If $(M, N) \in h$ then there is i with $(M, N) = (\mathcal{C}(\sigma, x_i), \mathcal{C}(\rho, x_i)) = (\mathbf{e}(F_i\sigma), \mathbf{e}(F_i\rho))$.
- Else, $(M, N) = (\mathrm{E}_a(M'), \mathrm{E}_b(N'))$ and by the induction assumption there exists G such that $(M', N') = (\mathbf{e}(G\sigma), \mathbf{e}(G\rho))$. By Lemma 2.4.1, $(a, b) \in h$, so there exists i such that $(a, b) = (\mathcal{C}(\sigma, x_i), \mathcal{C}(\rho, x_i)) = (\mathbf{e}(F_i\sigma), \mathbf{e}(F_i\rho))$. But then $(M, N) = (\mathrm{E}_{\mathbf{e}(F_i\sigma)}(\mathbf{e}(G\sigma)), \mathrm{E}_{\mathbf{e}(F_i\rho)}(\mathbf{e}(G\rho))) = (\mathbf{e}(\mathrm{E}_{F_i}(G)\,\sigma), \mathbf{e}(\mathrm{E}_{F_i}(G)\,\rho))$.

□

We now show that an environment represented by a consistent pair of substitutions can accept a given message pair iff an environment represented by a hedge can do it, and characterize the hedge derived under φ.

Lemma 2.4.42 *If* $\sigma\{^M/_x\} \cong \rho\{^N/_x\}$ *then* $\mathcal{I}(h \cup \{(M, N)\}) = \varphi(\sigma\{^M/_x\}, \rho\{^N/_x\})$.

Proof. Let $g = \{(\sigma(x_i), \rho(x_i)) | x_i \in \mathrm{dom}(\sigma)\}$. By Lemma 2.4.15 we have $\mathcal{I}(\mathcal{I}(g) \cup \{(M, N)\}) = \mathcal{I}(g \cup \{(M, N)\})$. By using Lemma 2.4.37 to (σ, ρ) we get that $\mathcal{I}(g) = h$, and substitution gives that $\mathcal{I}(h \cup \{(M, N)\}) = \mathcal{I}(g \cup \{(M, N)\})$. Applying Lemma 2.4.37 to $(\sigma\{^M/_x\}, \rho\{^N/_x\})$ we get $\mathcal{I}(g \cup \{(M, N)\}) = \varphi(\sigma\{^M/_x\}, \rho\{^N/_x\})$.

Thus $\mathcal{I}(h \cup \{(M, N)\}) = \varphi(\sigma\{^M/_x\}, \rho\{^N/_x\})$, as desired. □

Lemma 2.4.43 *If* $\mathcal{I}(h \cup \{(M, N)\})$ *is consistent, then* $\sigma\{^M/_x\} \cong \rho\{^N/_x\}$ *for any* $x \notin \mathrm{dom}(\sigma)$.

Proof. Let $g = \mathcal{I}(h \cup \{(M, N)\})$. By Lemma 2.4.39 $g_1^f \cong g_2^f$. Take any injection $f' : \mathrm{dom}(g_1^f) \to \mathcal{N}$ such that $\mathrm{range}(f') \cap (\mathrm{dom}(g_1^f) \cup \mathrm{dom}(\sigma) \cup \{x\}) = \emptyset$. We define $\overline{g}_i^f = \{^{g_i^f(x_1)}/_{f'(x_1)}, \ldots, ^{g_i^f(x_n)}/_{f'(x_n)}\}_{x_j \in \mathrm{dom}(g_1^f)}$. By iterated application of Lemma 2.4.30 we have that $g_1^f \circ \overline{g}_1^f \cong g_2^f \circ \overline{g}_2^f$, and Lemma 2.4.31 gives that $\overline{g}_1^f \cong \overline{g}_2^f$. Moreover, since $\{\, (g_1^f(x_i), g_2^f(x_i)) \mid x_i \in \mathrm{dom}(g_1^f) \,\} = \{\, (\overline{g}_1^f(x_i), \overline{g}_2^f(x_i)) \mid x_i \in \mathrm{dom}(\overline{g}_1^f) \,\}$, Lemma 2.4.37 gives that $g = \varphi(\overline{g}_1^f, \overline{g}_2^f)$.

By Lemma 2.4.40 we have that $\forall x_i \in \mathrm{dom}(\sigma) : h \vdash \sigma(x_i) \leftrightarrow \rho(x_i)$. By Lemma 2.4.10 we have that $h \cup \{(M,N)\} \leq g$, so $g \vdash \sigma(x_i) \leftrightarrow \rho(x_i)$. By Lemma 2.4.41 $(\overline{g}_1^f, \overline{g}_2^f) \vdash \sigma(x_i) \leftrightarrow \rho(x_i)$, so by definition there are F_i such that $\mathbf{e}(F_i\overline{g}_1^f) = \sigma(x_i)$, $\mathbf{e}(F_i\overline{g}_2^f) = \rho(x_i)$ and $\mathrm{n}(F_i) \subseteq \mathrm{dom}(\overline{g}_1^f)$.

Then, by iterated application of Lemma 2.4.30 we get that $\overline{g}_1^f \circ \sigma \cong \overline{g}_2^f \circ \rho$. Similarly, as $g \vdash M \leftrightarrow N$ we have by Lemma 2.4.41 that $(\overline{g}_1^f, \overline{g}_2^f) \vdash M \leftrightarrow N$. By definition this means there is a G such that $\mathrm{n}(G) \subseteq \mathrm{dom}(\overline{g}_1^f)$, $\mathbf{e}(G\overline{g}_1^f) = M$ and $\mathbf{e}(G\overline{g}_2^f) = N$. By Lemma 2.4.30 $\overline{g}_1^f \circ \sigma\{^M/_x\} \cong \overline{g}_2^f \circ \rho\{^N/_x\}$. Finally, by iterated application of Lemma 2.4.31 $\sigma\{^M/_x\} \cong \rho\{^N/_x\}$. □

We also need to see what happens when the environment creates fresh names.

Lemma 2.4.44 *If $B = \{b_1, b_2, \ldots, b_n\} \subset \mathcal{N}$ is finite and $B \cap \mathrm{fn}(\sigma,\rho) = \emptyset$ then $h \cup \mathrm{Id}_B = \varphi(\sigma\{^{b_1}/_{y_1}, \ldots, ^{b_n}/_{y_n}\}, \rho\{^{b_1}/_{y_1}, \ldots, ^{b_n}/_{y_n}\})$ for all sets of variables $Y = \{y_1, y_2, \ldots, y_n\}$ such that $Y \cap \mathrm{range}(\sigma) = \emptyset$.*

Proof. We write $\{^B/_Y\}$ for $\{^{b_1}/_{y_1}, \ldots, ^{b_n}/_{y_n}\}$. By Lemma 2.4.35 we have that h is consistent, so since B is fresh we have that $h \cup \mathrm{Id}_B$ is consistent by Lemma 2.4.20. By Lemma 2.4.43 we get that $\sigma\{^B/_Y\} \cong \rho\{^B/_Y\}$.

Since $b_i \in \mathcal{N}$ we have that $\mathcal{C}(\sigma\{^B/_Y\}, b_i) = b_i$ and $\mathcal{C}(\rho\{^B/_Y\}, b_i) = b_i$. Since B is fresh we have for any $x_i \in \mathrm{dom}(\sigma)$ that $\mathcal{C}(\sigma\{^B/_Y\}, x_i) = \mathcal{C}(\sigma, x_i)$ and $\mathcal{C}(\rho\{^B/_Y\}, x_i) = \mathcal{C}(\rho, x_i)$, so $\sigma\{^B/_Y\} \cong \rho\{^B/_Y\}$ and $\varphi(\sigma\{^B/_Y\}, \rho\{^B/_Y\}) = h \cup \mathrm{Id}_B$. □

We use the preceding result to show a variant of Lemma 2.4.41, namely that if a hedge augmented with some fresh names can create a pair of messages then the same pair of messages can be produced by a corresponding consistent pair of substitutions. Note that in the definition of alley bisimilarity, the fresh names are required to occur in the expressions, while the other bisimilarities allow the creation of fresh names that are not used.

Lemma 2.4.45 *If B is a finite set of names such that $B \cap \mathrm{fn}(\sigma,\rho) = \emptyset$ and $h \cup \mathrm{Id}_B \vdash M \leftrightarrow N$ then there exists F such that $\mathrm{n}(F) \setminus \mathrm{dom}(\sigma) = B$ and $\mathbf{e}(F\sigma) = M, \mathbf{e}(F\rho) = N$.*

Proof. $h \cup \mathrm{Id}_B = \varphi(\sigma\{^{b_1}/_{y_1}, \ldots, ^{b_n}/_{y_n}\}, \rho\{^{b_1}/_{y_1}, \ldots, ^{b_n}/_{y_n}\})$ by Lemma 2.4.44. Then, according to Lemma 2.4.41 $(\sigma\{^{b_1}/_{y_1}, \ldots, ^{b_n}/_{y_n}\}, \rho\{^{b_1}/_{y_1}, \ldots, ^{b_n}/_{y_n}\}) \vdash M \leftrightarrow N$, so there is $G : \mathrm{n}(G) \subseteq \mathrm{dom}(\sigma) \cup B$ and $\mathbf{e}(G\sigma) = M, \mathbf{e}(G\rho) = N$. Clearly, $F = \mathrm{D}_{b_1}(\mathrm{D}_{b_2}(\cdots \mathrm{D}_{b_n}(\mathrm{E}_{b_n}(\cdots \mathrm{E}_{b_2}(\mathrm{E}_{b_1}(G))\cdots))\cdots))$ has the desired properties. □

2.4.5 Frames and Alleys

The missing component in our relations between the various kinds of environments concerns the relation between frame-theory pairs and substitution pairs. As a matter of fact, we are only interested in the case of strongly consistent substitution pairs. This definition is in terms of φ, as defined in Definition 2.4.34.

Definition 2.4.46 *The frame-theory pair corresponding to a (strongly consistent) substitution pair is defined by:*

$$\gamma : \mathbf{SS} \to \mathbf{FT} : \gamma(\sigma, \rho) := (\ \mathcal{N} \cap \pi_1(\varphi(\sigma, \rho))\ ,\ \varphi(\sigma, \rho) \setminus (\mathcal{N} \times \mathcal{N})\ )$$

Note that this definition indeed only makes sense in the case that $\sigma \cong_{\mathrm{s}} \rho$, because in it we consider for the frame only the names of $\pi_1(\varphi(\sigma, \rho))$ while ignoring those of $\pi_2(\varphi(\sigma, \rho))$. Note further that all of the properties we need later on for γ will be derived from the properties of φ.

2.4.6 Message Equivalence

As we saw in Section 2.3, two consistent non-blind environments can be $(\sim_{\mathrm{x}}, \sim_{\mathrm{y}})$-equivalent only if they are $\mathcal{M}$-equivalent (see Definition 2.3.9). With the results of this section, we can fully characterize $\mathcal{M}$-equivalent frame-theory pairs and alleys and, because of this, give necessary conditions on full abstractions on the corresponding bisimilarities.

As showed in Lemma 2.4.18, two $\mathcal{M}$-equivalent consistent hedges are equal. Using this, we can show that $\mathcal{M}$-equivalent consistent frame-theory pairs are equal. First we need to show that ψ is injective when restricted to the set of consistent frame-theory pairs.

Lemma 2.4.47 *If* $(\mathrm{fr}, \mathrm{th})$ *and* $(\mathrm{fr}', \mathrm{th}')$ *are consistent and* $\psi(\mathrm{fr}, \mathrm{th}) = \psi(\mathrm{fr}', \mathrm{th}')$ *then* $(\mathrm{fr}, \mathrm{th}) = (\mathrm{fr}', \mathrm{th}')$.

Proof. Let $h = \psi(\mathrm{fr}, \mathrm{th}) = \psi(\mathrm{fr}', \mathrm{th}')$. By definition, $(M, N) \in h$ iff $M = N \in \mathrm{fr}$, $M = N \in \mathrm{fr}'$, $(M, N) \in \mathrm{th}$ or $(M, N) \in \mathrm{th}'$.

Take $(M, N) \in h$. If M or N is a name then $M = N \in \mathrm{fr}$ and $M = N \in \mathrm{fr}'$ by Lemma 2.4.22. Otherwise we must have $(M, N) \in \mathrm{th}$ and $(M, N) \in \mathrm{th}'$, since the frame only contains names. □

Two $\mathcal{M}$-equivalent consistent frame-theory pairs are equal.

Lemma 2.4.48 *If* $(\mathrm{fr}, \mathrm{th})$ *and* $(\mathrm{fr}', \mathrm{th}')$ *are consistent and* $(\mathrm{fr}, \mathrm{th}) \gtrless (\mathrm{fr}', \mathrm{th}')$ *then* $(\mathrm{fr}, \mathrm{th}) = (\mathrm{fr}', \mathrm{th}')$.

Proof. Note that $\psi(\mathrm{fr}, \mathrm{th}) \gtrless \psi(\mathrm{fr}', \mathrm{th}')$. $\psi(\mathrm{fr}, \mathrm{th})$ and $\psi(\mathrm{fr}', \mathrm{th}')$ are both consistent by Lemma 2.4.23, so $\psi(\mathrm{fr}, \mathrm{th}) = \psi(\mathrm{fr}', \mathrm{th}')$ by Lemma 2.4.18. Then $(\mathrm{fr}, \mathrm{th}) = (\mathrm{fr}', \mathrm{th}')$ by Lemma 2.4.47. □

Corollary 2.4.49 *Lemma 2.4.48 together with Proposition 2.3.11 give that all full abstractions from* [*framed, fenced*] *to* [*framed, fenced*] *must be the identity on consistent, non-blind environments.*

$\mathcal{M}$-equivalent pairs of substitutions are exactly those that correspond to the same hedge.

Lemma 2.4.50 $(\sigma, \rho) \gtrless (\sigma', \rho')$ *if and only if* $\varphi(\sigma, \rho) = \varphi(\sigma', \rho')$.

Proof. By Lemma 2.4.40 and Lemma 2.4.41 we have that $(\sigma, \rho) \gtrless (\sigma', \rho')$ iff $\varphi(\sigma, \rho) \gtrless \varphi(\sigma', \rho')$. $\varphi(\sigma, \rho)$ and $\varphi(\sigma', \rho')$ are consistent by Lemma 2.4.35, so Lemma 2.4.18 gives that $\varphi(\sigma, \rho) \gtrless \varphi(\sigma', \rho')$ iff $\varphi(\sigma, \rho) = \varphi(\sigma', \rho')$. □

Corollary 2.4.51 *Lemma 2.4.50 together with Proposition 2.3.11 give that all full abstractions from* [*alley, trellis*] *to* [*alley, trellis*] *must preserve the image under* φ *of all consistent, non-blind environments.*

2.5 Comparing Bisimulations

Having established the relations between the different kinds of environment representations, we may study the relations between the respective bisimulations. In general, this is done by lifting the environment mapping functions to consistent environmental relations. In general the lifting is defined as follows:

Definition 2.5.1 *Assume that* $g : \mathbf{E}_{\mathrm{x}} \to \mathbf{E}_{\mathrm{y}}$ *is an environment mapping.*
If e_{x} *is consistent for* $\sim_{\mathrm{x}}$ *we let* $G(e_{\mathrm{x}}, P, Q) := (g(e_{\mathrm{x}}), P, Q)$.
If $\mathcal{R}$ *is a consistent relation (for* $\sim_{\mathrm{x}}$*) we let* $G(\mathcal{R}) := \{\, G(e_{\mathrm{x}}, P, Q) \mid e_{\mathrm{x}} \vdash P \;\mathcal{R}\; Q \,\}$.
If $\mathcal{S}$ *is a consistent relation (for* $\sim_{\mathrm{y}}$*) we let* $G^{-1}(\mathcal{S}) := \{(e_{\mathrm{x}}, P, Q) \mid G(e_{\mathrm{x}}, P, Q) \in \mathcal{S}\}$.
By abuse of notation, we write $G^{-1}(e_{\mathrm{y}}, P, Q)$ *for* $G^{-1}(\{(e_{\mathrm{y}}, P, Q)\})$.

For instance, recalling γ from Definition 2.4.46 we have $\Gamma(\mathcal{R}) = \{(\gamma(\sigma, \rho), P, Q) \mid ((\sigma, \rho), P, Q) \in \mathcal{R}\}$ whenever $\mathcal{R}$ is a strongly consistent alley relation.

2.5.1 Fenced vs. Trellis Bisimulation

These two bisimulations were compared by [FHJ01]. Here, we recapitulate and add to their results. We use the environment mapping γ and its lifting Γ, as instances of the above definition scheme.

These are the main results of [FHJ01]:

Theorem 2.5.2 *$\Gamma(\sim_{\mathrm{s}})$ is a fenced bisimulation.*

Theorem 2.5.3 *$\Gamma^{-1}(\sim_{\#})$ is a strongly consistent alley bisimulation.*

Using the terminology of Definition 2.3.1, these results give us the following Lemma.

Lemma 2.5.4 *$\sim_{\#}$ is fully γ-abstract w.r.t. $\sim_{\mathrm{s}}$.*

Proof. Fix $\sigma \cong_{\mathrm{s}} \rho$ and take any P, Q. We need to check that $(\sigma, \rho) \vdash P \sim_{\mathrm{s}} Q$ if and only if $\gamma(\sigma, \rho) \vdash P \sim_{\#} Q$. By Theorem 2.5.3 we have the "if" and Theorem 2.5.2 gives the "only if". □

Intuitively, this means that $\sim_{\mathrm{s}}$ can be embedded in $\sim_{\#}$. However, this is only one of the full abstractions needed to prove fenced and trellis equivalent. For the other direction, we must find an environment mapping going in the opposite direction of γ. The composition of ψ (Definition 2.4.21) and θ^f (Definition 2.4.38) is a reasonable candidate.

Lemma 2.5.5

1. *If* $(\mathrm{fr}, \mathrm{th})$ *is consistent then* $\theta_1^f(\psi(\mathrm{fr}, \mathrm{th})) \cong_{\mathrm{s}} \theta_2^f(\psi(\mathrm{fr}, \mathrm{th}))$.
2. $\gamma \circ \theta^f \circ \psi = \mathrm{Id}$ *on the set of consistent frame-theory pairs.*

Proof. $\psi(\mathrm{fr}, \mathrm{th})$ is consistent by Lemma 2.4.23, so $\theta_1^f(\psi(\mathrm{fr}, \mathrm{th})) \cong \theta_1^f(\psi(\mathrm{fr}, \mathrm{th}))$ according to Lemma 2.4.39. Since $(\mathrm{fr}, \mathrm{th})$ is consistent, the only pairs of names in $\psi(\mathrm{fr}, \mathrm{th})$ are of the type (a, a) where $a \in \mathrm{fr}$. This gives that we actually have $\theta_1^f(\psi(\mathrm{fr}, \mathrm{th})) \cong_{\mathrm{s}} \theta_1^f(\psi(\mathrm{fr}, \mathrm{th}))$.

By Lemma 2.4.39 we also have that $\varphi(\theta^f(\psi(\mathrm{fr}, \mathrm{th}))) = \psi(\mathrm{fr}, \mathrm{th})$. Since $(\mathrm{fr}, \mathrm{th})$ is consistent by assumption, we have that $\mathrm{fr} = \mathcal{N} \cap \pi_1(\psi(\mathrm{fr}, \mathrm{th}))$ and $\mathrm{th} = \psi(\mathrm{fr}, \mathrm{th}) \setminus (\mathcal{N} \times \mathcal{N})$. □

Lemma 2.5.6 *$\sim_{\mathrm{s}}$ is fully $\theta^f \circ \psi$-abstract w.r.t. $\sim_{\#}$.*

Proof. Fix a consistent frame-theory pair $(\mathrm{fr}, \mathrm{th})$, and take any P, Q. We need to check that $(\mathrm{fr}, \mathrm{th}) \vdash P \sim_{\#} Q$ iff $\theta^f(\psi(\mathrm{fr}, \mathrm{th})) \vdash P \sim_{\mathrm{s}} Q$.

We have by Lemma 2.5.5 that $\Gamma(\theta^f(\psi(\mathrm{fr}, \mathrm{th})), P, Q) = ((\mathrm{fr}, \mathrm{th}), P, Q)$. Using this, Theorem 2.5.2 gives the "if" and the "only if" follows from Theorem 2.5.3. □

By combining Lemma 2.5.6 and Lemma 2.5.4, we have the desired equivalence.

Theorem 2.5.7 *$\sim_{\mathrm{s}}$ is $(\gamma, \theta^f \circ \psi)$-equivalent to $\sim_{\#}$.*

2.5.2 Fenced vs. Hedged Bisimulation

The correspondence between fenced and hedged bisimulation is easy to find, since they have similar environments and, according to Lemma 2.4.27, the same action on process output. On the level of the environments the correspondence is given by ψ, and we lift this to Ψ according to the above definition scheme.

We immediately get an embedding of fenced bisimulation in hedged.

Theorem 2.5.8 *$\Psi(\sim_\#)$ is a hedged bisimulation.*

Proof. $\Psi(\sim_\#)$ is a consistent hedged relation by Lemma 2.4.23 and is symmetric by the symmetry of $\sim_\#$. Assume that $(\mathrm{fr}, \mathrm{th}) \vdash P \sim_\# Q$.

1. If $P \xrightarrow{\tau} P'$ then there is a Q' such that $Q \xrightarrow{\tau} Q'$ and $(\mathrm{fr}, \mathrm{th}) \vdash P' \sim_\# Q'$, so $(\psi(\mathrm{fr}, \mathrm{th}), P', Q') \in \Psi(\sim_\#)$.
2. Assume that $P \xrightarrow{a(x)} P'$, $\psi(\mathrm{fr}, \mathrm{th}) \vdash a \leftrightarrow b$ and $B \subset \mathcal{N}$ is finite such that $B \cap (\mathrm{fn}(P, Q) \cup \mathrm{n}(\psi(\mathrm{fr}, \mathrm{th}))) = \emptyset$ and $\psi(\mathrm{fr}, \mathrm{th}) \cup \mathrm{Id}_B \vdash M \leftrightarrow N$. By Lemma 2.4.1 $(a, b) \in \psi(\mathrm{fr}, \mathrm{th})$ and as $(\mathrm{fr}, \mathrm{th})$ is consistent we get $a = b \in \mathrm{fr}$. Clearly $\mathrm{n}(\psi(\mathrm{fr}, \mathrm{th})) = \mathrm{fr} \cup \mathrm{n}(\mathrm{th})$. Then there is Q' such that $Q \xrightarrow{a(x)} Q'$ and $(\mathrm{fr} \cup B, \mathrm{th}) \vdash P'\{^M/_x\} \sim_\# Q'\{^N/_x\}$, so $(\psi(\mathrm{fr}, \mathrm{th}) \cup \mathrm{Id}_B, P'\{^M/_x\}, Q'\{^N/_x\}) \in \Psi(\sim_\#)$.
3. Assume that $P \xrightarrow{(\nu \tilde{c})\,\overline{a}\,M} P'$, $\psi(\mathrm{fr}, \mathrm{th}) \vdash a \leftrightarrow b$ and $\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{n}(\pi_1(h))) = \emptyset$. As above $a = b \in \mathrm{fr}$ and $\mathrm{n}(\pi_1(\psi(\mathrm{fr}, \mathrm{th}))) = \mathrm{fr} \cup \mathrm{n}(\pi_1(\mathrm{th}))$ so there are $Q', N, \tilde{d}$ where $Q \xrightarrow{(\nu \tilde{d})\,\overline{b}\,N} Q'$, $(\mathrm{fn}(Q) \cup \mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th}))) \cap \{\tilde{d}\} = \emptyset$ and $\xi(\mathrm{fr}, \mathrm{th}, M, N) \vdash P' \sim_\# Q'$. Clearly $\mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th})) = \mathrm{n}(\pi_2(\psi(\mathrm{fr}, \mathrm{th})))$. $\psi(\xi(\mathrm{fr}, \mathrm{th}, M, N)) = \mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\})$ by Lemma 2.4.27, so $(\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\}), P', Q') \in \Psi(\sim_\#)$.

□

We may also state this as

Corollary 2.5.9 *$\sim_\mathrm{h}$ is ψ-complete w.r.t. $\sim_\#$.*

Note that ψ satisfies the conditions we highlighted in Section 2.3, namely that it preserves knowledge, consistency and non-blindness. However, as seen in Section 2.2.1, ψ does not yield a full abstraction.

2.5.3 Fenced vs. Framed Bisimulation

One of the main results of [EHHO99] is the following:

Theorem 2.5.10 *If $(\mathrm{fr}, \mathrm{th}) \vdash P \sim_\# Q$ then $(\mathrm{fr}, \mathrm{th}) \vdash P \sim_\mathrm{f} Q$.*

The authors enunciate that the converse would also hold, but our counterexamples in Section 2.2 show that this cannot be the case. Indeed, their proof is flawed[3]. However, Theorem 2.5.10 can be restated as

Corollary 2.5.11 *Let Id be the identity mapping on frame-theory pairs.*

- $\sim_{\mathrm{f}}$ *is Id-complete w.r.t.* $\sim_{\#}$.
- $\sim_{\#}$ *is Id-sound w.r.t.* $\sim_{\mathrm{f}}$.

By transitivity, we also get

Corollary 2.5.12

- $\sim_{\mathrm{f}}$ *is* γ*-complete w.r.t.* $\sim_{\mathrm{s}}$.
- $\sim_{\mathrm{s}}$ *is* $\theta^f \circ \psi$*-sound w.r.t.* $\sim_{\mathrm{f}}$.

Also here, the examples in Section 2.2 show that neither of Id, γ and $\theta^f \circ \psi$ yield a full abstraction.

2.5.4 Framed vs. Hedged Bisimulation

In framed bisimulation we can make arbitrary extensions to the environment on process output, something that is not permitted in hedged bisimulation. Therefore, we can only show a direct correspondence between framed bisimilarity and hedged bisimilarity up to weakening. We need a lemma concerning extensions of frame-theory pairs and their corresponding hedges.

Lemma 2.5.13 *If* $(\mathrm{fr}', \mathrm{th}')$ *is consistent,* $(\mathrm{fr}, \mathrm{th}) \leq (\mathrm{fr}', \mathrm{th}')$ *and* $(\mathrm{fr}', \mathrm{th}') \vdash M \leftrightarrow N$ *then* $\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\})$ *is consistent and* $\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\}) \leq \psi(\mathrm{fr}', \mathrm{th}')$.

Proof. By Corollary 2.4.5(2) $\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\} \leq \psi(\mathrm{fr}', \mathrm{th}')$. This gives by Lemma 2.4.13 that $\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\}) \leq \mathcal{I}(\psi(\mathrm{fr}', \mathrm{th}'))$. $\psi(\mathrm{fr}', \mathrm{th}')$ is consistent by Lemma 2.4.23, so $\psi(\mathrm{fr}', \mathrm{th}') = \mathcal{I}(\psi(\mathrm{fr}', \mathrm{th}'))$ according to Lemma 2.4.16.

$\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\})$ is irreducible by Corollary 2.4.8, so the consistency follows from Lemma 2.4.19 applied to $\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\}) \leq \psi(\mathrm{fr}', \mathrm{th}')$. □

Theorem 2.5.14 $\Psi(\sim_{\mathrm{f}})$ *is a hedged bisimulation up to weakening.*

[3] In the proof of their Lemma 20, stating (roughly) soundness of framed up to weakening, actions that must be simulated by the minimal environment but not by a larger one are not taken into account (cf. the example in Section 2.2).

Proof. Assume that $(\mathrm{fr}, \mathrm{th}) \vdash P \sim_{\mathrm{f}} Q$. Internal calculation and input is handled as in the proof of Theorem 2.5.8.

Assume that $P \xrightarrow{(\nu \tilde{c})\,\overline{a}\,M} P'$, $\psi(\mathrm{fr}, \mathrm{th}) \vdash a \leftrightarrow b$ and $\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{n}(\pi_1(h))) = \emptyset$. By Lemma 2.4.22 we have $a = b \in \mathrm{fr}$.

$\mathrm{n}(\pi_1(\psi(\mathrm{fr}, \mathrm{th}))) = \mathrm{fr} \cup \mathrm{n}(\pi_1(\mathrm{th}))$ so there are $Q', N, \tilde{d}, \mathrm{fr}', \mathrm{th}'$ with $Q \xrightarrow{(\nu \tilde{d})\,\overline{b}\,N} Q'$, $(\mathrm{fn}(Q) \cup \mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th}))) \cap \{\tilde{d}\} = \emptyset$, $(\mathrm{fr}, \mathrm{th}) \leq (\mathrm{fr}', \mathrm{th}')$, $(\mathrm{fr}', \mathrm{th}') \vdash M \leftrightarrow N$ and $(\mathrm{fr}', \mathrm{th}') \vdash P' \sim_{\mathrm{f}} Q'$. Moreover $\mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th})) = \mathrm{n}(\pi_2(\psi(\mathrm{fr}, \mathrm{th})))$.

We need to show that there is $h' \geq \mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\})$ such that $h' \vdash P'\ \Psi(\sim_{\mathrm{f}})\ Q'$. By Lemma 2.5.13 we have that $\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\})$ is consistent and $\mathcal{I}(\psi(\mathrm{fr}, \mathrm{th}) \cup \{(M, N)\}) \leq \psi(\mathrm{fr}', \mathrm{th}')$, and since $(\mathrm{fr}', \mathrm{th}') \vdash P' \sim_{\mathrm{f}} Q'$ (see above) we get that $\psi(\mathrm{fr}', \mathrm{th}') \vdash P'\ \Psi(\sim_{\mathrm{f}})\ Q'$. □

In proving this theorem, we needed to weaken the hedges on process output. Actually, since fenced bisimilarity is not complete with respect to framed, construction of a non-minimal environment may be *required* to get a framed bisimulation. Since $\sim_{\mathrm{h}}$ only performs minimal extensions, we get that $\Psi(\sim_{\mathrm{f}})$ is not a hedged bisimulation.

We now proceed to show that up to weakening is a sound proof technique for hedged bisimulation. The standard proof for this is to show that for an arbitrary hedged bisimulation up to weakening $\mathcal{R}$ we have that $\mathcal{R}_{\mathrm{w}}$ is a hedged bisimulation. However, this is in fact not the case, as this example shows.

Example 2.5.15 *We let*

$$\begin{aligned}\mathcal{R} \quad \stackrel{\mathrm{def}}{=} \quad & \{(\{(a,a), (\mathrm{E}_k(a), \mathrm{E}_k(a))\}, (\nu l)\,\overline{a}\langle l\rangle.\,\mathbf{0}, (\nu l)\,\overline{a}\langle l\rangle.\,\mathbf{0})\} \\ & \cup \{(\{(a,a), (\mathrm{E}_k(a), \mathrm{E}_k(a)), (l,l)\}, \mathbf{0}, \mathbf{0}) \mid l \neq a, k\}.\end{aligned}$$

Then $\mathcal{R}$ is a hedged bisimulation and thus a hedged bisimulation up to weakening, but $\mathcal{R}_{\mathrm{w}}$ is not a hedged bisimulation similarly to Example 2.1.27.

We instead show that "up to bijective renaming and weakening" is sound, which using commutativity and Proposition 2.1.26 yields the soundness of up to weakening. We first give a simpler definition of up to weakening for consistent relations.

Lemma 2.5.16 *If R is a consistent hedged relation then $h \vdash P\ \mathcal{R}_{\mathrm{w}}\ Q$ iff h is irreducible and there is h' such that $h' \vdash P\ \mathcal{R}\ Q$ and $h \leq h'$.*

Proof: By Definition 2.4.2 $h \leq h'$ iff $\mathcal{S}(h) \subseteq \mathcal{S}(h')$. We will use the $\leq$ notation throughout this proof.

Assume that $h \vdash P\ \mathcal{R}_{\mathrm{w}}\ Q$. By definition, there are $\overline{h}, h'$ such that $h = \mathcal{I}(\overline{h})$, $h' \vdash P\ \mathcal{R}\ Q$ and $\overline{h} \leq h'$. By Corollary 2.4.8 h is irreducible. Lemma 2.4.13 gives

that $h \leq \mathcal{I}(h')$, but h' is consistent by the consistency of $\mathcal{R}$ so $h' = \mathcal{I}(h')$ by Lemma 2.4.16. Thus $h \leq h'$.

Assume that h is irreducible and there is h' such that $h' \vdash P \mathcal{R} Q$ and $h \leq h'$. By Definition 2.4.6 $h = \mathcal{I}(h)$, so $h \vdash P \mathcal{R}_{\mathrm{w}} Q$. □

Theorem 2.5.17 *Hedged bisimulation is sound up to bijective renaming and weakening.*

PROOF: Let $\mathcal{R}$ be a hedged bisimulation up to bijective renaming and weakening. We wish to show that $\mathcal{R} \subseteq \sim_{\mathrm{h}}$. We do this by showing that $\mathcal{R}_{\mathrm{bw}}$ is a hedged bisimulation; the result follows since $\mathcal{R} \subseteq \mathcal{R}_{\mathrm{bw}}$ because b and w are both expansions.

If σ is a bijective substitution $\mathcal{N} \to \mathcal{N}$ we write σ^{-1} for its inverse. Since $\mathcal{R}$ is consistent, we use the alternative definition of $\mathcal{R}_{\mathrm{w}}$ due to Lemma 2.5.16.

Assume that $h \vdash P \mathcal{R}_{\mathrm{bw}} Q$ and that $P \xrightarrow{\mu} P'$. By the definition of $\mathcal{R}_{\mathrm{bw}}$, h is irreducible and there are $\overline{h}, \overline{P}, \overline{Q}$ and bijective $\sigma : \mathcal{N} \to \mathcal{N}$ and $\rho : \mathcal{N} \to \mathcal{N}$ such that $\overline{h} \vdash \overline{P} \mathcal{R} \overline{Q}$, $\overline{P}\sigma = P$, $\overline{Q}\rho = Q$, and $h \leq \overline{h}(\sigma, \rho)$.

1. Assume that $\mu = \tau$.
   Since σ is bijective $\overline{P} \xrightarrow{\tau} P'\sigma^{-1}$. Since $\mathcal{R}$ is a hedged bisimulation up to bijective renaming and weakening there is Q' such that $\overline{Q} \xrightarrow{\tau} Q'$ and $\overline{h} \vdash P'\sigma^{-1} \mathcal{R}_{\mathrm{bw}} Q'$. This gives that $h \vdash P' \mathcal{R}_{\mathrm{bwb}} Q'\rho$. Since $\mathcal{R}_{\mathrm{bwb}} = \mathcal{R}_{\mathrm{bw}}$ we actually have $h \vdash P' \mathcal{R}_{\mathrm{bw}} Q'\rho$. We also have that $Q \xrightarrow{\tau} Q'\rho$ since ρ is bijective, so Q can simulate the transition $P \xrightarrow{\mu} P'$.
2. Assume that $\mu = a(x)$, $h \vdash a \leftrightarrow b$ and $B \subset \mathcal{N}$ is finite such that $h \cup \mathrm{Id}_B \vdash M \leftrightarrow N$ and $B \cap (\mathrm{fn}(P, Q) \cup \mathrm{n}(h)) = \emptyset$.
   Since B are not necessarily fresh for $\overline{h}, \overline{P}, \overline{Q}$, we need to invent another set of fresh names. Take $C \subset \mathcal{N}$ with $|C| = |B|$ and $C \cap (\mathrm{n}(\overline{h}, h) \cup \mathrm{fn}(\overline{P}, \overline{Q}, P, Q)) = \emptyset$. Let σ', ρ' be bijective substitutions such that $\sigma'(C) = C = \rho'(C)$, $\sigma'|_{\mathrm{n}(\pi_1(\overline{h})) \cup \mathrm{fn}(\overline{P})} = \sigma|_{\mathrm{n}(\pi_1(\overline{h})) \cup \mathrm{fn}(\overline{P})}$ and $\rho'|_{\mathrm{n}(\pi_2(\overline{h})) \cup \mathrm{fn}(\overline{Q})} = \rho|_{\mathrm{n}(\pi_2(\overline{h})) \cup \mathrm{fn}(\overline{Q})}$. Let $\eta : B \to C$ be a bijective function.
   In this case, $P = \overline{P}\sigma'$, $Q = \overline{Q}\rho'$, $P = P\eta$, $Q = Q\eta$, $h \cup \mathrm{Id}_C \vdash M\eta \leftrightarrow N\eta$ and $h \leq \overline{h}(\sigma', \rho')$. Moreover, $P \xrightarrow{a\, M\eta} P'\eta$. Since σ' is bijective, we then get that $\overline{P} \xrightarrow{(a\sigma'^{-1})\, M\eta\sigma'^{-1}} P'\eta\sigma'^{-1}$. Since σ' is bijective, we then get that $\overline{P} \xrightarrow{(a\sigma'^{-1})\, M\eta\sigma'^{-1}} P'\sigma'^{-1}$.
   Since $h \leq \overline{h}(\sigma', \rho')$ we have $\overline{h} \vdash (a\sigma'^{-1}) \leftrightarrow (b\rho'^{-1}b)$ and $\overline{h} \cup \mathrm{Id}_C \vdash M\eta\sigma'^{-1} \leftrightarrow N\eta\rho'^{-1}$. Since $\mathcal{R}$ is a hedged bisimulation up to bijective renaming and weakening there is Q' such that $\overline{Q} \xrightarrow{(b\rho'^{-1})(x)} Q'$ and $\overline{h} \cup \mathrm{Id}_C \vdash P'\{^x/_M\}\eta\sigma'^{-1} \mathcal{R}_{\mathrm{bw}} Q'\{^x/_{N\eta\rho'^{-1}}\}$.
   By Corollary 2.4.5.3 we have $h \cup \mathrm{Id}_C \leq \overline{h} \cup \mathrm{Id}_C$. Moreover, since $\sigma'\eta^{-1}$ and $\rho'\eta^{-1}$ are bijective, we get that $h \cup \mathrm{Id}_B \vdash P' \mathcal{R}_{\mathrm{bwbw}} Q'\rho'\eta^{-1}$. As above,

$\mathcal{R}_{\text{bwbw}} = \mathcal{R}_{\text{bw}}$, and Q can simulate the transition $P \xrightarrow{a(x)} P'$ since $Q \xrightarrow{b(x)} Q'\rho'$.

3. Assume that $\mu = (\nu\tilde{c})\,\overline{a}\,M$, $h \vdash a \leftrightarrow b$ and $\{\tilde{c}\} \cap (\text{fn}(P) \cup \text{n}(\pi_1(h))) = \emptyset$.
   Since σ is bijective, $\overline{P} \xrightarrow{(\nu\tilde{c}\sigma^{-1})\,\overline{(a\sigma^{-1})}\,M\sigma^{-1}} P'\sigma^{-1}$. Since $h \leq \overline{h}(\sigma,\rho)$ we have that $\overline{h} \vdash (a\sigma^{-1}) \leftrightarrow (b\rho^{-1})$. Since $\mathcal{R}$ is a hedged bisimulation up to bijective renaming and weakening we have that there are $Q', \tilde{d}, N$ such that $\overline{Q} \xrightarrow{(\nu\tilde{d})\,\overline{(b\rho^{-1})}\,N} Q'$ with $\{\tilde{d}\} \cap (\text{fn}(\overline{Q}) \cup \text{n}(\pi_2(\overline{h}))) = \emptyset$ and $\mathcal{I}(\overline{h} \cup \{(M\sigma^{-1}, N)\}) \vdash P'\sigma^{-1}\ \mathcal{R}_{\text{bw}}\ Q'$. By Corollary 2.4.5.3 we get that $h \cup \{(M, N\rho)\} \leq \overline{h}(\sigma,\rho) \cup \{(M, N\rho)\}$. Lemma 2.4.13 then gives that $\mathcal{I}(h \cup \{(M, N\rho)\}) \leq \mathcal{I}(\overline{h}(\sigma,\rho) \cup \{(M, N\rho)\})$, so $\mathcal{I}(h \cup \{(M, N\rho)\}) \vdash P'\ \mathcal{R}_{\text{bwbw}}\ Q'\rho$, where $\mathcal{R}_{\text{bwbw}} = \mathcal{R}_{\text{bw}}$ as above. Since ρ is bijective we get $Q \xrightarrow{(\nu\tilde{d}\rho)\,\overline{b}\,N\rho} Q'\rho$, so Q simulates the transition $P \xrightarrow{\mu} P'$.

□

Corollary 2.5.18 *Hedged bisimulation is sound up to weakening.*

Together with Corollary 2.5.18, this gives the desired completeness result.

Corollary 2.5.19 $\sim_{\text{h}}$ *is* ψ*-complete w.r.t.* $\sim_{\text{f}}$.

Together with Proposition 2.1.25, we also get the following corollary.

Corollary 2.5.20 *Let* $g \leq h$ *be irreducible. If* $h \vdash P \sim_{\text{h}} Q$ *then* $g \vdash P \sim_{\text{h}} Q$.

2.5.5 Alley vs. Hedged bisimulation

Now it is time to show the equivalence between alley bisimilarity and hedged bisimilarity. Recall the environment mappings φ (Definition 2.4.34) and θ^f (Definition 2.4.38). We lift φ to Φ according to the definition schema above.

Theorem 2.5.21 $\Phi(\sim_{\text{a}})$ *is a hedged bisimulation.*

Proof. $\Phi(\sim_{\text{a}})$ is symmetric by the symmetry of $\sim_{\text{a}}$ and consistent by Lemma 2.4.35. Assume that $(\sigma,\rho) \vdash P \sim_{\text{a}} Q$ and that $h = \varphi(\sigma,\rho)$.

1. If $P \xrightarrow{\tau} P'$ there exists Q' such that $Q \xrightarrow{\tau} Q'$ and $(\sigma,\rho) \vdash P' \sim_{\text{a}} Q'$, so $\Phi((\sigma,\rho), P', Q') = (h, P', Q') \in \Phi(\sim_{\text{a}})$.
2. Assume that $P \xrightarrow{a(x)} P'$, $h \vdash a \leftrightarrow b$ and $B \subset \mathcal{N}$ is finite such that $B \cap (\text{fn}(P,Q) \cup \text{n}(h)) = \emptyset$ and $h \cup \text{Id}_B \vdash M \leftrightarrow N$.
   By Lemma 2.4.41 we get that $(\sigma,\rho) \vdash a \leftrightarrow b$. By Lemma 2.4.45 there is G such that $\text{n}(G) \setminus \text{dom}(\sigma) = B$ and $\mathbf{e}(G\sigma) = M, \mathbf{e}(G\rho) = N$.

As $(\sigma,\rho) \vdash P \sim_{\mathrm{a}} Q$ there exist $Q', \tilde{y}$ such that $Q \xrightarrow{b(x)} Q'$ and $(\sigma\{^{b_1}/_{y_1},\ldots,^{b_n}/_{y_n}\}, \rho\{^{b_1}/_{y_1},\ldots,^{b_n}/_{y_n}\}) \vdash P'\{^{M}/_{x}\} \sim_{\mathrm{a}} Q'\{^{N}/_{x}\}$.
Let $\{^{B}/_{Y}\} = \{^{b_1}/_{y_1},\ldots,^{b_n}/_{y_n}\}$.
By Lemma 2.4.44 we get that $h \cup \mathrm{Id}_B = \varphi(\sigma\{^{B}/_{Y}\}, \rho\{^{B}/_{Y}\})$, so
$\Phi((\sigma\{^{B}/_{Y}\}, \rho\{^{B}/_{Y}\}), P', Q') = (h \cup \mathrm{Id}_B, P'\{^{M}/_{x}\}, Q'\{^{N}/_{x}\}) \in \Phi(\sim_{\mathrm{a}})$.

3. Assume that $P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} P', h \vdash a \leftrightarrow b$ and $(\mathrm{fn}(P) \cup \mathrm{n}(\pi_1(h))) \cap \{\tilde{c}\} = \emptyset$.
By Lemma 2.4.41 we get $(\sigma,\rho) \vdash a \leftrightarrow b$. By Corollary 2.4.36 $\mathrm{n}(\pi_1(h)) = \mathrm{fn}(\sigma)$, so $\mathrm{fn}(P,\sigma) \cap \{\tilde{c}\} = \emptyset$. As $(\sigma,\rho) \vdash P \sim_{\mathrm{a}} Q$ there exists $Q', N, \tilde{d}$ such that $Q \xrightarrow{(\nu\tilde{d})\,\overline{b}\,N} Q'$, $\mathrm{fn}(Q,\rho) \cap \{\tilde{d}\} = \emptyset$ and $(\sigma\{^{M}/_{x}\}, \rho\{^{N}/_{x}\}) \vdash P' \sim_{\mathrm{a}} Q'$.
By Corollary 2.4.36 $\mathrm{fn}(\rho) = \mathrm{n}(\pi_2(h))$, so $(\mathrm{fn}(Q) \cup \mathrm{n}(\pi_2(h))) \cap \{\tilde{d}\} = \emptyset$. According to Lemma 2.4.42 $\varphi(\sigma\{^{M}/_{x}\}, \rho\{^{N}/_{x}\}) = \mathcal{I}(h \cup \{(M,N)\})$, so
$\Phi((\sigma\{^{M}/_{x}\}, \rho\{^{N}/_{x}\}), P', Q') = (\mathcal{I}(h \cup \{(M,N)\}), P', Q') \in \Phi(\sim_{\mathrm{a}})$.

□

Theorem 2.5.22 *$\Phi^{-1}(\sim_{\mathrm{h}})$ is an alley bisimulation.*

Proof. $\Phi^{-1}(\sim_{\mathrm{h}})$ is symmetric by the symmetry of $\sim_{\mathrm{h}}$. As Φ is only defined for consistent pairs of substitutions $\Phi^{-1}(\sim_{\mathrm{h}})$ is consistent. Assume that $h \vdash P \sim_{\mathrm{h}} Q$ where $h = \varphi(\sigma,\rho)$.

1. If $P \xrightarrow{\tau} P'$ then there exists Q' such that $Q \xrightarrow{\tau} Q'$ and $h \vdash P' \sim_{\mathrm{h}} Q'$. Clearly $((\sigma,\rho), P', Q') \in \Phi^{-1}(h, P', Q')$.
2. Assume that $(\sigma,\rho) \vdash a \leftrightarrow b$ and that $B = \mathrm{n}(G) \setminus \mathrm{dom}(\sigma)$ is such that $B \cap \mathrm{fn}(P,Q,\rho,\sigma) = \emptyset$. Furthermore, let $\mathbf{e}(G\sigma) = M, \mathbf{e}(G\rho) = N$ and assume that $P \xrightarrow{a(x)} P'$. Assume that $|B| = n$ and let $Y = \{c_1, c_2, \ldots, c_n\}$ be any set of names not in $\mathrm{dom}(\sigma)$. We write $\{^{B}/_{Y}\}$ for $\{^{b_1}/_{y_1},\ldots,^{b_n}/_{y_n}\}$.
By Lemma 2.4.44 we have that $h \cup \mathrm{Id}_B = \varphi(\sigma\{^{B}/_{Y}\}, \rho\{^{B}/_{Y}\})$. We also get that $h \vdash a \leftrightarrow b$ and $h \cup \mathrm{Id}_B \vdash M \leftrightarrow N$ by Lemma 2.4.40. Now, as $h \vdash P \sim_{\mathrm{h}} Q$ there is Q' such that $Q \xrightarrow{b(x)} Q'$ and $h \cup \mathrm{Id}_B \vdash P'\{^{M}/_{x}\} \sim_{\mathrm{h}} Q'\{^{N}/_{x}\}$. Note that $\Phi((\sigma\{^{B}/_{Y}\}, \rho\{^{B}/_{Y}\}), P'\{^{M}/_{x}\}, Q'\{^{N}/_{x}\}) = (h \cup \mathrm{Id}_B, P'\{^{M}/_{x}\}, Q'\{^{N}/_{x}\})$.
3. Assume that $(\sigma,\rho) \vdash a \leftrightarrow b$ and $P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} P'$ with $\mathrm{fn}(P,\sigma) \cap \{\tilde{c}\} = \emptyset$. By Lemma 2.4.40 we have that $h \vdash a \leftrightarrow b$, and by Corollary 2.4.36 $\mathrm{fn}(\sigma) = \mathrm{n}(\pi_1(h))$. As $h \vdash P \sim_{\mathrm{h}} Q$ there are $Q', N, \tilde{d}$ such that $Q \xrightarrow{(\nu\tilde{d})\,\overline{b}\,N} Q'$, $(\mathrm{fn}(Q) \cup \mathrm{n}(\pi_2(h))) \cap \{\tilde{d}\} = \emptyset$ and $\mathcal{I}(h \cup \{(M,N)\}) \vdash P' \sim_{\mathrm{h}} Q'$. By Corollary 2.4.36 $\mathrm{fn}(\rho) = \mathrm{n}(\pi_2(h))$, so $\mathrm{fn}(Q,\rho) \cap \{\tilde{d}\} = \emptyset$.
Then $\mathcal{I}(h \cup \{(M,N)\}) = \varphi(\sigma\{^{M}/_{x}\}, \rho\{^{N}/_{x}\})$ by Lemma 2.4.42, so
$\Phi((\sigma\{^{M}/_{x}\}, \rho\{^{N}/_{x}\}), P', Q') = (\mathcal{I}(h \cup \{(M,N)\}), P', Q')$.

□

We now restate these results using the terminology of Definition 2.3.1.

Lemma 2.5.23 *$\sim_{\mathrm{h}}$ is fully φ-abstract w.r.t. $\sim_{\mathrm{a}}$.*

Proof. Fix $\sigma \cong \rho$, and take any P, Q. We need to check that $(\sigma, \rho) \vdash P \sim_{\mathrm{a}} Q$ if and only if $\varphi(\sigma, \rho) \vdash P \sim_{\mathrm{h}} Q$. By Theorem 2.5.22 we have the "if" and Theorem 2.5.21 gives the "only if". □

Lemma 2.5.24 *$\sim_{\mathrm{a}}$ is fully θ^f-abstract w.r.t. $\sim_{\mathrm{h}}$.*

Proof. Fix a consistent hedge h. We need to check that for all processes P, Q we have that $h \vdash P \sim_{\mathrm{h}} Q$ if and only if $\theta^f(h) \vdash P \sim_{\mathrm{a}} Q$. By Lemma 2.4.39 we have that $\Phi(\theta^f(h), P, Q) = (h, P, Q)$. Using this, Theorem 2.5.21 gives the "if" and the "only if" follows from Theorem 2.5.22. □

By combining these results, we have

Theorem 2.5.25 *$\sim_{\mathrm{a}}$ is (φ, θ^f)-equivalent to $\sim_{\mathrm{h}}$.*

We can use this theorem to transfer results on hedged bisimilarity to alleys. For instance, we have that $\mathcal{M}$-equivalent alleys can be substituted for each other in bisimulations.

Lemma 2.5.26 *If $(\sigma, \rho) \gtrless (\sigma', \rho')$ then $(\sigma, \rho) \vdash P \sim_{\mathrm{a}} Q$ iff $(\sigma', \rho') \vdash P \sim_{\mathrm{a}} Q$*

Proof. By Lemma 2.4.50 we have that $\varphi(\sigma, \rho) = \varphi(\sigma', \rho')$. The result follows from Lemma 2.5.23 which states that $(\sigma, \rho) \vdash P \sim_{\mathrm{a}} Q$ iff $\varphi(\sigma, \rho) \vdash P \sim_{\mathrm{h}} Q$. □

2.5.6 Negative Results

Having found the full abstractions above, we can now disprove the existence of full abstractions between other pairs of bisimilarities.

Proposition 2.5.27 *In this proposition, we write "$\sim_{\mathrm{x}}$ is not fully abstract w.r.t. $\sim_{\mathrm{y}}$" for "there is no mapping G such that $\sim_{\mathrm{x}}$ is fully G-abstract w.r.t. $\sim_{\mathrm{y}}$".*

1. *$\sim_{\mathrm{f}}$ is not fully abstract w.r.t. $\sim_{\mathrm{h}}$, $\sim_{\#}$, $\sim_{\mathrm{a}}$ or $\sim_{\mathrm{s}}$.*
2. *$\sim_{\#}$ is not fully abstract w.r.t. $\sim_{\mathrm{h}}$, $\sim_{\mathrm{f}}$ or $\sim_{\mathrm{a}}$.*
3. *$\sim_{\mathrm{h}}$ is not fully abstract w.r.t. $\sim_{\mathrm{f}}$, $\sim_{\#}$ or $\sim_{\mathrm{s}}$.*
4. *$\sim_{\mathrm{a}}$ is not fully abstract w.r.t. $\sim_{\mathrm{f}}$, $\sim_{\#}$ or $\sim_{\mathrm{s}}$.*
5. *$\sim_{\mathrm{s}}$ is not fully abstract w.r.t. $\sim_{\mathrm{f}}$, $\sim_{\mathrm{h}}$ or $\sim_{\mathrm{a}}$.*

Proof. To show that there is no function $g : \mathbf{E}_{\mathrm{x}} \to \mathbf{E}_{\mathrm{y}}$ satisfying $e_{\mathrm{x}} \equiv^{\mathrm{x}}_{\mathrm{y}} g(e_{\mathrm{x}})$, we show that there is one consistent non-blind environment e_{x} that is not $(\sim_{\mathrm{x}}, \sim_{\mathrm{y}})$-equivalent to any of its $\mathcal{M}$-equivalent counterparts in $\mathbf{E}_{\mathrm{y}}$. Alternatively, we may use the transitivity of full abstractness to derive a contradiction.

1. (a) $h := \{(a, a)\}$ is consistent but not h-blind, since $h \vdash \overline{a}\langle a\rangle.\,\mathbf{0} \not\sim_{\mathrm{h}} \mathbf{0}$. We look for a $(\sim_{\mathrm{h}}, \sim_{\mathrm{f}})$-equivalent frame-theory pair $(\mathrm{fr}, \mathrm{th})$. Proposition 2.3.11 then gives that such a frame-theory pair must be consistent and satisfy $(\mathrm{fr}, \mathrm{th}) \geqslant h$.
   Since $(\mathrm{fr}, \mathrm{th})$ is consistent we have by Lemma 2.4.23 that $\psi(\mathrm{fr}, \mathrm{th})$ is consistent. We also have that $(\mathrm{fr}, \mathrm{th}) \geqslant \psi(\mathrm{fr}, \mathrm{th})$, so the transitivity of $\geqslant$ gives that $\psi(\mathrm{fr}, \mathrm{th}) \geqslant h$. Since both of these hedges are consistent, Lemma 2.4.18 gives that $\psi(\mathrm{fr}, \mathrm{th}) = h$. Then Lemma 2.4.22 gives us that $(\mathrm{fr}, \mathrm{th}) = (\{a\}, \emptyset)$, which has been shown in Section 2.2.2 not to relate the same processes as the hedge h.
   (b) By Proposition 2.2.4 there exists a consistent non-blind $(\mathrm{fr}, \mathrm{th})$ such that $(\mathrm{fr}, \mathrm{th}) \not\equiv^{\mathrm{f}}_{\#} (\mathrm{fr}, \mathrm{th})$. We try to find a frame-theory pair $(\mathrm{fr}', \mathrm{th}')$ that is $(\sim_{\mathrm{f}}, \sim_{\#})$-equivalent to $(\mathrm{fr}, \mathrm{th})$.
   By Proposition 2.3.11 we have that such a frame-theory pair must be consistent and non-blind, and that $(\mathrm{fr}', \mathrm{th}') \geqslant (\mathrm{fr}, \mathrm{th})$ must hold. Lemma 2.4.48 then gives that $(\mathrm{fr}', \mathrm{th}') = (\mathrm{fr}, \mathrm{th})$, but $(\mathrm{fr}, \mathrm{th}) \not\equiv^{\mathrm{f}}_{\#} (\mathrm{fr}, \mathrm{th})$.
   (c) Assume that $\sim_{\mathrm{f}}$ is fully g-abstract w.r.t. $\sim_{\mathrm{a}}$. We have by Lemma 2.5.24 that $\sim_{\mathrm{a}}$ is fully θ^f-abstract w.r.t. $\sim_{\mathrm{h}}$. By transitivity this implies that $\sim_{\mathrm{f}}$ is fully $g \circ \theta^f$-abstract w.r.t. $\sim_{\mathrm{h}}$, which contradicts 1(a).
   (d) Assume that $\sim_{\mathrm{f}}$ is fully g-abstract w.r.t. $\sim_{\mathrm{s}}$. We have by Lemma 2.5.6 that $\sim_{\mathrm{s}}$ is fully $\theta^f \circ \psi$-abstract w.r.t. $\sim_{\#}$. By transitivity this implies that $\sim_{\mathrm{f}}$ is fully $g \circ \theta^f \circ \psi$-abstract w.r.t. $\sim_{\#}$, which contradicts 1(b).
2. As 1(a),(b),(c).
3. (a) By Proposition 2.2.12 there exists a consistent non-blind $(\mathrm{fr}, \mathrm{th})$ such that $(\mathrm{fr}, \mathrm{th}) \not\equiv^{\mathrm{f}}_{\mathrm{h}} \psi(\mathrm{fr}, \mathrm{th})$. We try to find a hedge h that is $(\sim_{\mathrm{f}}, \sim_{\mathrm{h}})$-equivalent to $(\mathrm{fr}, \mathrm{th})$.
   By Proposition 2.3.11 we have that such a hedge must be consistent and non-blind, and that $h \geqslant (\mathrm{fr}, \mathrm{th})$ must hold. Since $(\mathrm{fr}, \mathrm{th}) \geqslant \psi(\mathrm{fr}, \mathrm{th})$ we have by transitivity that $h \geqslant \psi(\mathrm{fr}, \mathrm{th})$. Note that $\psi(\mathrm{fr}, \mathrm{th})$ is consistent by Lemma 2.4.23. Then we can use Lemma 2.4.18 to derive that $h = \psi(\mathrm{fr}, \mathrm{th})$, which is a contradiction.
   (b) As 3(a).
   (c) As 1(d).
4. (a) Assume that $\sim_{\mathrm{a}}$ is fully g-abstract w.r.t. $\sim_{\mathrm{f}}$. We have by Lemma 2.5.23 that $\sim_{\mathrm{h}}$ is fully φ-abstract w.r.t. $\sim_{\mathrm{a}}$. By transitivity this implies that $\sim_{\mathrm{h}}$ is fully $\varphi \circ g$-abstract w.r.t. $\sim_{\mathrm{f}}$, which is false by 3(a).

	Alley	Hedged	Framed	Fenced	Trellis	$\sim_\top$	$\sim_\bot$
Alley		F	s,C	s,C	s,C	F	F
Hedged	F		s,C	s,C	s,C	F	F
Framed	s,c	s,c		s,C	s,C	F	F
Fenced	s,c	s,c	S,c		F	F	F
Trellis	s,c	s,c	S,c	F		F	F
$\sim_\top$	c	c	c	c	c		c
$\sim_\bot$	s	s	s	s	s	s	

Table 2.2: Relations Between the Bisimilarities.

(b) As 4(a).
(c) As 1(d).

5. (a) Assume that $\sim_{\mathrm{s}}$ is fully g-abstract w.r.t. $\sim_{\mathrm{f}}$. We have by Lemma 2.5.4 that $\sim_\#$ is fully γ-abstract w.r.t. $\sim_{\mathrm{s}}$. By transitivity this implies that $\sim_\#$ is fully $\gamma \circ g$-abstract with respect to $\sim_{\mathrm{f}}$, which is false by 2(b).
(b) As 5(a).
(c) As 5(a).

□

We summarize the relations between the bisimilarities in Table 2.2. Following the discussion in Section 2.3, we consider environment mappings that preserve synthesis, consistency and non-blindness as "good". In the table, an "s" means that there is a trivial function g making the bisimilarity leading the row sound with respect to the one heading the column and "c" stands for trivial completeness. A "S" means that there is a "good" sound environment mapping, "C" stands for the existence of a "good" complete one and "F" for any full abstraction. We don't show relations that are subsumed by stronger ones. Note that there are no environment mappings from alleys or hedges to frame-theory pairs preserving both soundness and the synthesis (cf. Lemma 2.4.22).

2.6 Comparison in a Categorical Framework

Usually, bisimilarities are represented and studied as sets of process pairs; comparisons between bisimilarities are therefore based on set-theoretic comparisons. In the previous sections, we have achieved a set-theoretic understanding of the relations between the various environment-sensitive bisimilarities, properly taking into account the relations (represented by mappings) between the environment components. It turns out that we can improve on this merely set-theoretic understanding and fur-

ther refine the comparison of the bisimilarities by not only studying the elements of bisimilarities, but also the (pairs of matching) transitions that connect them. The application of this technique to bisimilarities for other process calculi (e.g., the pi calculus) is a subject of possible future work.

As we have seen, definitions of bisimulations typically contain statements such as "If $e \vdash P \mathcal{R} Q$ and $P \xrightarrow{\tau} P'$ then there exists Q' such that $Q \xrightarrow{\tau} Q'$ and $e \vdash P' \mathcal{R} Q'$". This provides us with some internal structure on the set $\mathcal{R}$, namely transition pairs connecting the objects (e, P, Q) and (e, P', Q'). For bookkeeping purposes, we label such arrows with the process actions. This procedure, more formally defined below, turns each bisimulation itself into a labelled transition system, which allows us to study the internal structure of the bisimilarities. As a standard uniform framework for this kind of comparison, we use the language of category theory. We straightforwardly redefine the bisimilarities as categories, then we lift the environment mappings to functors between those categories and study the properties of these functors.

Let $\mathcal{A}$ denote the set of *actions* defined by the following grammar:

$$\mu, \gamma \quad ::= \quad \tau \quad | \quad \overline{a}\, M \quad | \quad a\, M$$

Note that new names are not explicitly mentioned on process output. However, by inspection of the derivation of $P \xrightarrow{(\nu \tilde{c})\, \overline{a}\, M} P'$ we have that $\{\tilde{c}\} \subseteq \mathrm{n}(M)$. (Note that $\tilde{c} = \mathrm{n}(M) \setminus \mathrm{fn}(P)$ may be false, due to the rules Sum, Let and Guard.) Since the simulations only consider outputs where $\tilde{c}$ is fresh (see Section 2.1 for the precise meaning in each particular case), we have that $\tilde{c}$ is simply the fresh names in $\mathrm{n}(M)$.

2.6.1 Redefinitions

The categorical definitions below all have a similar structure:

- The objects are process pairs under some kind of environment. Processes are considered up to α-equivalence.
- The arrows are labelled, and correspond to matching transitions of the process pair. Due to the closure requirement on the composition of arrows in a category, the labels must be strings over the set of actions.
- Composition of arrows is concatenation of the labels, which clearly is associative. If $A \xrightarrow{\tilde{\mu}} B$ and $B \xrightarrow{\tilde{\gamma}} C$ then we write $A \xrightarrow{\tilde{\mu}\tilde{\gamma}} C$ for the arrow $B \xrightarrow{\tilde{\gamma}} C \circ A \xrightarrow{\tilde{\mu}} B$.
- Two arrows are considered equal if they have the same label, domain and codomain.
- The identity arrows simply correspond to not doing any transition, and are labelled with the empty string (different from τ, since identity arrows must be idempotent and $\tau\tau \neq \tau$ as strings).

Note that the arrows are only labelled with the actions of the first process in the pair. Together with the codomain, this uniquely determines the actions of the second process, since a given consistent environment can not consider a given message equivalent to two different messages. We may now proceed to the redefinitions.

Framed Bisimulation

The category $\mathfrak{F}$ has $\sim_{\mathrm{f}}$ as set of objects. We say that there is a primitive $\mathfrak{F}$-arrow from $((\mathrm{fr},\mathrm{th}),P,Q)$ to $((\mathrm{fr}',\mathrm{th}'),P',Q')$ iff one of the following conditions holds:

1. $P \xrightarrow{\tau} P'$, $Q \xrightarrow{\tau} Q'$ and $(\mathrm{fr}',\mathrm{th}') = (\mathrm{fr},\mathrm{th})$. This arrow is labelled with τ.
2. $P \xrightarrow{a(x)} P''$, $Q \xrightarrow{a(x)} Q''$, $\mathrm{th}' = \mathrm{th}$ and $\mathrm{fr}' = \mathrm{fr} \cup B$ where $a \in \mathrm{fr}$, $B \subset \mathcal{N}$ is finite, $B \cap (\mathrm{fn}(P,Q) \cup \mathrm{fr} \cup \mathrm{n}(\mathrm{th})) = \emptyset$, $P' = P''\{^{M}/_{x}\}$, $Q' = Q''\{^{N}/_{x}\}$ and $(\mathrm{fr} \cup B, \mathrm{th}) \vdash M \leftrightarrow N$. This arrow is labelled with $a\,M$.
3. $P \xrightarrow{(\nu \tilde{c})\,\overline{a}\,M} P'$, $Q \xrightarrow{(\nu \tilde{d})\,\overline{a}\,N} Q'$, $(\mathrm{fr},\mathrm{th}) \leq (\mathrm{fr}',\mathrm{th}')$ and $(\mathrm{fr}',\mathrm{th}') \vdash M \leftrightarrow N$ where $a \in \mathrm{fr}$,$\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{fr} \cup \mathrm{n}(\pi_1(\mathrm{th}))) = \emptyset$ and $\{\tilde{d}\} \cap (\mathrm{fn}(Q) \cup \mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th}))) = \emptyset$. This arrow is labelled with $\overline{a}\,M$.

The arrows in $\mathfrak{F}$ are the identity arrows and the transitive closure of the primitive $\mathfrak{F}$-arrows.

Fenced Bisimulation

The category $\mathfrak{F}_{\#}$ has $\sim_{\#}$ as set of objects. We say that there is a primitive $\mathfrak{F}_{\#}$-arrow from $((\mathrm{fr},\mathrm{th}),P,Q)$ to $((\mathrm{fr}',\mathrm{th}'),P',Q')$ iff one of the following conditions holds:

1. $P \xrightarrow{\tau} P'$, $Q \xrightarrow{\tau} Q'$ and $(\mathrm{fr}',\mathrm{th}') = (\mathrm{fr},\mathrm{th})$. This arrow is labelled with τ.
2. $P \xrightarrow{a(x)} P''$, $Q \xrightarrow{a(x)} Q''$, $\mathrm{th}' = \mathrm{th}$ and $\mathrm{fr}' = \mathrm{fr} \cup B$, where $a \in \mathrm{fr}$, $B \subset \mathcal{N}$ is finite, $B \cap (\mathrm{fn}(P,Q) \cup \mathrm{fr} \cup \mathrm{n}(\mathrm{th})) = \emptyset$, $P' = P''\{^{M}/_{x}\}$, $Q' = Q''\{^{N}/_{x}\}$ and $(\mathrm{fr} \cup B, \mathrm{th}) \vdash M \leftrightarrow N$. This arrow is labelled with $a\,M$.
3. $P \xrightarrow{(\nu \tilde{c})\,\overline{a}\,M} P'$, $Q \xrightarrow{(\nu \tilde{d})\,\overline{a}\,N} Q'$ and $(\mathrm{fr}',\mathrm{th}') = \xi(\mathrm{fr},\mathrm{th},M,N)$ where $a \in \mathrm{fr}$, $\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{fr} \cup \mathrm{n}(\pi_1(\mathrm{th}))) = \emptyset$ and $\{\tilde{d}\} \cap (\mathrm{fn}(Q) \cup \mathrm{fr} \cup \mathrm{n}(\pi_2(\mathrm{th}))) = \emptyset$. This arrow is labelled with $\overline{a}\,M$.

The arrows in $\mathfrak{F}_{\#}$ are the identity arrows and the transitive closure of the primitive $\mathfrak{F}_{\#}$-arrows.

Alley Bisimulation

The category $\mathfrak{A}$ has $\sim_{\mathrm{a}}$ as set of objects. We say that there is a primitive $\mathfrak{A}$-arrow from $((\sigma,\rho),P,Q)$ to $((\sigma',\rho'),P',Q')$ iff one of the following conditions holds:

1. $P \xrightarrow{\tau} P'$, $Q \xrightarrow{\tau} Q'$, $\sigma' = \sigma$ and $\rho' = \rho$. This arrow is labelled with τ.
2. $P \xrightarrow{a(x)} P''$, $Q \xrightarrow{b(x)} Q''$, $\sigma' = \sigma\{^B/_Y\}$ and $\rho' = \rho\{^B/_Y\}$ where $B \cap \mathrm{fn}(P, Q, \rho, \sigma) = \emptyset$, $Y \cap \mathrm{dom}(\sigma) = \emptyset$, $(\sigma, \rho) \vdash a \leftrightarrow b$, $P' = P''\{^M/_x\}$, $Q' = Q''\{^N/_x\}$ and there exists G such that $B = \mathrm{fn}(G) \setminus \mathrm{dom}(\sigma)$, $\mathbf{e}(G\sigma) = M$ and $\mathbf{e}(G\rho) = N$. This arrow is labelled with $a\,M$.
3. $P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} P'$, $Q \xrightarrow{(\nu\tilde{d})\,\overline{b}\,N} Q'$, $\sigma' = \sigma\{^M/_x\}$ and $\rho' = \rho\{^N/_x\}$ where $\mathrm{fn}(P, \sigma) \cap \{\tilde{c}\} = \emptyset$, $\mathrm{fn}(Q, \rho) \cap \{\tilde{d}\} = \emptyset$ and $(\sigma, \rho) \vdash a \leftrightarrow b$. This arrow is labelled with $\overline{a}\,M$.

The arrows in $\mathfrak{A}$ are the identity arrows and the transitive closure of the primitive $\mathfrak{A}$-arrows.

Trellis Bisimulation

The category $\mathfrak{S}$ is the sub-category of $\mathfrak{A}$ obtained by restricting the set of objects to $\sim_{\mathrm{s}}$.

Hedged Bisimulation

The category $\mathfrak{H}$ has $\sim_{\mathrm{h}}$ as set of objects. We say that there is a primitive $\mathfrak{H}$-arrow from (h, P, Q) to (h', P', Q') iff one of the following conditions holds:

1. $P \xrightarrow{\tau} P'$, $Q \xrightarrow{\tau} Q'$ and $h' = h$. This arrow is labelled with τ.
2. $P \xrightarrow{a(x)} P''$, $Q \xrightarrow{b(x)} Q''$ and $h' = h \cup \mathrm{Id}_B$ where $h \vdash a \leftrightarrow b$, $h \cup \mathrm{Id}_B \vdash M \leftrightarrow N$, $P' = P''\{^M/_x\}$, $Q' = Q''\{^N/_x\}$, $B \subset \mathcal{N}$ is finite and $B \cap (\mathrm{fn}(P, Q) \cup \mathrm{n}(h)) = \emptyset$. This arrow is labelled with $a\,M$.
3. $P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} P'$, $Q \xrightarrow{(\nu\tilde{d})\,\overline{b}\,N} Q'$ and $h' = \mathcal{I}(h \cup \{(M, N)\})$ where $h \vdash a \leftrightarrow b$, $\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{n}(\pi_1(h))) = \emptyset$ and $\{\tilde{d}\} \cap (\mathrm{fn}(Q) \cup \mathrm{n}(\pi_2(h))) = \emptyset$. This arrow is labelled with $\overline{a}\,M$.

The arrows of $\mathfrak{H}$ are the identity arrows and the transitive closure of the primitive $\mathfrak{H}$-arrows.

2.6.2 Reinterpretation

We now attempt to lift our environment mappings to functors between the bisimilarities, and study the properties of these functors.

Framed and Fenced

We begin by comparing $\mathfrak{F}$ and $\mathfrak{F}_\#$. Regarding the objects, $\sim_\# \subsetneq \sim_\mathrm{f}$ by Proposition 2.2.4 and Theorem 2.5.10. At process output we have that $((\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\overline{a}\,M} ((\mathrm{fr}',\mathrm{th}'),P',Q')$ in $\mathfrak{F}_\#$ only if $(\mathrm{fr}',\mathrm{th}') = \xi(\mathrm{fr},\mathrm{th},M,N)$. In $\mathfrak{F}$, we also have an arrow $\xrightarrow{\overline{a}\,M}$ from $((\mathrm{fr},\mathrm{th}),P,Q)$ to $(\xi(\mathrm{fr},\mathrm{th},M,N),P',Q')$ according to Lemma 2.4.25 and Theorem 2.5.10(1). As framed and fenced bisimulations behave identically on process input and internal calculation, there is a trivial embedding functor $\mathfrak{F}_\# \to \mathfrak{F}$. However, in $\mathfrak{F}$ we are allowed to further extend the frame-theory pair on process output, giving rise to arrows not present in $\mathfrak{F}_\#$.

Example 2.6.1 *Let $P = \overline{a}\langle a\rangle.\,\mathbf{0}$. Then $(\{a\},\emptyset) \vdash P \sim_\# P$ and $(\{a\},\emptyset) \vdash P \sim_\mathrm{f} P$. Except for the identity, the only arrow from $((\{a\},\emptyset),P,P)$ in $\mathfrak{F}_\#$ is $((\{a\},\emptyset),P,P) \xrightarrow{\overline{a}\,a} ((\{a\},\emptyset),\mathbf{0},\mathbf{0})$. However, in $\mathfrak{F}$ there are arrows $((\{a\},\emptyset),P,P) \xrightarrow{\overline{a}\,a} ((\mathrm{fr},\mathrm{th}),\mathbf{0},\mathbf{0})$ whenever $(\mathrm{fr},\mathrm{th})$ is consistent and $a \in \mathrm{fr}$.*

Fenced and Hedged

We define $\Psi^\# : \mathfrak{F}_\# \to \mathfrak{H}$ as $\Psi^\#((\mathrm{fr},\mathrm{th}),P,Q) = (\psi(\mathrm{fr},\mathrm{th}),P,Q)$ and $\Psi^\#(((\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\tilde{\mu}} ((\mathrm{fr}',\mathrm{th}'),P',Q')) = (\psi(\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\tilde{\mu}} (\psi(\mathrm{fr}',\mathrm{th}'),P',Q')$. As shown in the proof of Theorem 2.5.8, any primitive $\mathfrak{F}_\#$-arrow corresponds to a primitive $\mathfrak{H}$-arrow with the same label, so by composing these we get that $\Psi^\#$ is a functor. $\Psi^\#$ is full, since both bisimilarities perform minimal extensions, and faithful, since arrows are equal iff they have the same labels, which are preserved by $\Psi^\#$. However, there is a hedged process pair in range($\Psi^\#$) with an $\mathfrak{H}$-arrow that leads outside range($\Psi^\#$).

Example 2.6.2 *Let $P = \overline{a}\langle a\rangle.(\nu l)\,\overline{a}\langle l\rangle.\,\mathbf{0} + \overline{a}\langle a\rangle.\overline{a}\langle k\rangle.\,\mathbf{0}$ where $k \neq a$. We have that $(\{a\},\emptyset) \vdash P \sim_\# P$, intuitively since the simulating process can always mimic the simulated. In $\mathfrak{H}$ we have that $(\{(a,a)\},P,P) \xrightarrow{\overline{a}\,a} (\{(a,a)\},(\nu l)\,\overline{a}\langle l\rangle.\,\mathbf{0},\overline{a}\langle k\rangle.\,\mathbf{0})$ but $(\{a\},\emptyset,(\nu l)\,\overline{a}\langle l\rangle.\,\mathbf{0},\overline{a}\langle k\rangle.\,\mathbf{0})$ is not in $\sim_\#$ by Proposition 2.2.10 and Theorem 2.5.10.*

We also have that the obvious extension of $\Psi^\#$ to $\mathfrak{F}$ is not a functor, because of the multitude of possible extensions of the frame-theory pair on process output (see Example 2.6.1).

Hedged and Alley

We extend the definition of Φ to $\mathfrak{A} \to \mathfrak{H}$ by letting $\Phi(((\sigma,\rho),P,Q) \xrightarrow{\tilde{\mu}} ((\sigma',\rho'),P',Q')) := (\varphi(\sigma,\rho),P,Q) \xrightarrow{\tilde{\mu}} (\varphi(\sigma',\rho'),P',Q')$. As shown in the proof of Theorem 2.5.8, any primitive $\mathfrak{F}_\#$-arrow corresponds to a primitive

$\mathfrak{H}$-arrow with the same label, so by composing these we get that Ψ is a functor. Ψ is faithful, since arrows are equal iff they have the same labels, which are preserved by Ψ. However, Ψ is not full, since several different non-isomorphic alley process pairs are mapped to a given hedged process pair.

We can also define $\Theta^f : \mathfrak{H} \to \mathfrak{A}$ by letting $\Theta^f(h, P, Q) := (\theta^f(h), P, Q)$ and $\Theta^f((h, P, Q) \xrightarrow{\tilde{\mu}} (h', P', Q')) := (\theta^f(h), P, Q) \xrightarrow{\tilde{\mu}} (\theta^f(h'), P', Q')$. Unfortunately Θ^f is not a functor, since on process output the environments in $\mathfrak{A}$ simply add message pairs instead of reducing them and discarding duplicates.

Example 2.6.3 *Let* $P = \overline{a}\langle \mathrm{E}_a(a)\rangle.\,\mathbf{0}$. *Clearly* $\{(a,a)\} \vdash P \sim_{\mathrm{h}} P$. *Assume that* $f(a,a) = x$ *where* f *is the function defining* θ^f. *As in the proof of Theorem 2.5.25 we have that* $(\{{}^{a}/_{x}\}, \{{}^{a}/_{x}\}) \vdash P \sim_{\mathrm{a}} P$. *In* $\mathfrak{H}$ *we have that* $(\{(a,a)\}, P, P) \xrightarrow{\overline{a}\,\mathrm{E}_a(a)} (\{(a,a)\}, \mathbf{0}, \mathbf{0})$. *A corresponding arrow in* $\mathfrak{A}$ *is* $((\{{}^{a}/_{x}\}, \{{}^{a}/_{x}\}), P, P) \xrightarrow{\overline{a}\,\mathrm{E}_a(a)} \left(\left(\{{}^{a}/_{x},\, {}^{\mathrm{E}_a(a)}/_{y}\}, \{{}^{a}/_{x},\, {}^{\mathrm{E}_a(a)}/_{y}\}\right), \mathbf{0}, \mathbf{0}\right)$. *However, we have that* $\Theta^f(\{(a,a)\}, \mathbf{0}, \mathbf{0}) = ((\{{}^{a}/_{x}\}, \{{}^{a}/_{x}\}), \mathbf{0}, \mathbf{0}) \neq \left(\left(\{{}^{a}/_{x},\, {}^{\mathrm{E}_a(a)}/_{y}\}, \{{}^{a}/_{x},\, {}^{\mathrm{E}_a(a)}/_{y}\}\right), \mathbf{0}, \mathbf{0}\right)$.

One way to fix this problem is the application of up-to techniques.

2.6.3 Up-to Techniques

The application of up-to techniques corresponds to adding up-to actions to $\mathcal{A}$. The following up-to techniques were defined for $\sim_{\mathrm{a}}$ by [BDP02].

- Up to structural congruence, which corresponds to adding s to the set of actions and adding all arrows $((\sigma,\rho), P, Q) \xrightarrow{s} ((\sigma,\rho), P', Q')$ where $P \equiv P'$ and $Q \equiv Q'$.
- Up to weakening, which corresponds to adding w to the set of actions and adding all arrows $((\sigma,\rho), P, Q) \xrightarrow{w} ((\sigma\{{}^{M}/_{x}\}, \rho\{{}^{N}/_{x}\}), P, Q)$.
- Up to contraction, which corresponds to adding c to the set of actions and adding all arrows $((\sigma\{{}^{M}/_{x}\}, \rho\{{}^{N}/_{x}\}), P, Q) \xrightarrow{c} ((\sigma,\rho), P, Q)$ where $(\sigma,\rho) \vdash M \leftrightarrow N$.
- Up to restriction, which corresponds to adding r to the set of actions and adding all arrows $((\sigma,\rho), (\nu\tilde{m})\,P, (\nu\tilde{n})\,Q) \xrightarrow{r} ((\sigma,\rho), P, Q)$ where $\{\tilde{m}\} \cap \mathrm{fn}(\sigma) = \emptyset$ and $\{\tilde{n}\} \cap \mathrm{fn}(\rho) = \emptyset$. When using this up-to technique it is preferable not to record the creation of fresh names on process output, since we want the output of a bound name to be equivalent to the composition of the lifting of the corresponding restriction and a non-bound output of the name.
- Up to parallel composition, which corresponds to adding p to the set of actions and adding all arrows $((\sigma,\rho), P \mid R\sigma, Q \mid R\rho) \xrightarrow{p} ((\sigma,\rho), P, Q)$ where $\mathrm{fn}(R) \subseteq \mathrm{dom}(\sigma)$.

The two equivalence relations on environments defined in Section 2.3 can also be used to define up-to techniques.

- For a given bisimilarity $\sim_{\mathrm{x}}$ we get the technique of up to $(\sim_{\mathrm{x}}, \sim_{\mathrm{x}})$-equivalence, which corresponds to adding an "up to $(\sim_{\mathrm{x}}, \sim_{\mathrm{x}})$-equivalence" action e to the set of actions and adding all arrows $(e_{\mathrm{x}}, P, Q) \xrightarrow{e} (e'_{\mathrm{x}}, P, Q)$ where $e_{\mathrm{x}} \equiv^{\mathrm{x}}_{\mathrm{x}} e'_{\mathrm{x}}$.
- Up to $\mathcal{M}$-equivalence, which means adding an "up to $\mathcal{M}$-equivalence" action m to the set of actions and adding all arrows $(e_{\mathrm{x}}, P, Q) \xrightarrow{m} (e'_{\mathrm{x}}, P, Q)$ where $e'_{\mathrm{x}} \gtrless e'_{\mathrm{x}}$.

After adding arrows, we must ensure that all compositions are defined. A priori, since we distinguish arrows having different labels we are sensitive to exactly where and which up-to techniques were used in the composition of an arrow, which is rarely desirable. However, we can make arrows that only differ in their use of up-to techniques equal by taking the quotient with the following equivalence:

Definition 2.6.4 *The arrows $A \xrightarrow{\tilde{\gamma}} B$ and $A \xrightarrow{\tilde{\mu}} B$ are up-to equivalent, written $\xrightarrow{\tilde{\gamma}} \equiv_u \xrightarrow{\tilde{\mu}}$ iff $\tilde{\gamma}\backslash U = \tilde{\mu}\backslash U$, i.e., $\tilde{\gamma}$ and $\tilde{\mu}$ are equal after removal of names of "up-to actions".*

In the resulting category, all arrows only labelled with reversible up-to actions are isomorphisms.

Using up-to techniques as defined above, we can give a more precise relation between framed and hedged bisimilarity expressed as categories.

Framed and Hedged

Apart from the environments the main difference between framed and hedged bisimulations is that we in framed bisimulation may extend the process pair more than strictly necessary on process output. Since we already have a mapping on the objects of the categories, to get an embedding it should intuitively suffice to permit arbitrary extensions also on the hedged side. This corresponds to taking hedged bisimilarity up to weakening.

We let $\mathfrak{H}^w$ be $\mathfrak{H}$ up to weakening and $\equiv_u$, i.e., the category obtained from $\mathfrak{H}$ by first adding all arrows of the type $(h, P, Q) \xrightarrow{w} (h', P, Q)$ where $h \leq h'$ and the codomain actually is an object in the category, then closing the set of arrows under composition and finally taking the quotient with $\equiv_u$ as defined in Definition 2.6.4. The effect of this is that we may add information to the hedge at any time — not only on process output! However, as we are only looking for an embedding of $\mathfrak{F}$ in $\mathfrak{H}^w$, the extra arrows resulting from our preference to use a standard up-to technique will not cause any problems.

We then define $\Psi^w : \mathfrak{F} \to \mathfrak{H}^w$ as $\Psi^w((\mathrm{fr}, \mathrm{th}), P, Q) = (\psi(\mathrm{fr}, \mathrm{th}), P, Q)$ on objects. On primitive $\mathfrak{F}$-arrows, Ψ^w acts in the following way.

1. We define $\Psi^w(((\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\tau} ((\mathrm{fr},\mathrm{th}),P',Q'))$ as the arrow $(\psi(\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\tau} (\psi(\mathrm{fr},\mathrm{th}),P',Q')$.
2. We define $\Psi^w(((\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{a\,M} ((\mathrm{fr}',\mathrm{th}'),P',Q'))$ as the arrow $(\psi(\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{a\,M} (\psi(\mathrm{fr}',\mathrm{th}'),P',Q'))$.
3. If $((\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\overline{a}\,M} ((\mathrm{fr}',\mathrm{th}'),P',Q')$ then by definition there is N such that $(\mathrm{fr}',\mathrm{th}') \vdash M \leftrightarrow N$. In $\mathfrak{H}$, there is an arrow $(\psi(\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\overline{a}\,M} (\mathcal{I}(\psi(\mathrm{fr},\mathrm{th})\cup\{(M,N)\}),P',Q')$, so we have to bridge the gap between the codomains of these arrows.
   We have $(\mathrm{fr},\mathrm{th}) \leq (\mathrm{fr}',\mathrm{th}')$ by the definition of framed bisimulation. By Lemma 2.5.13 we then get that $\mathcal{I}(\psi(\mathrm{fr},\mathrm{th})\cup\{(M,N)\}) \leq \psi((\mathrm{fr}',\mathrm{th}'))$, so there is a weakening arrow $(\mathcal{I}(\psi(\mathrm{fr},\mathrm{th}) \cup \{(M,N)\}),P',Q') \xrightarrow{w} (\psi((\mathrm{fr}',\mathrm{th}')),P',Q')$ bridging the gap. We then define $\Psi^w(((\mathrm{fr},\mathrm{th}),P,Q) \xrightarrow{\overline{a}\,M})((\mathrm{fr}',\mathrm{th}'),P',Q') := \xrightarrow{w} \circ \xrightarrow{\overline{a}\,M}$, where the domains and codomains of the arrows on the right are as seen above.

Inductively, the action of Ψ^w on composite $\mathfrak{F}$-arrows is just the composition of the application of Ψ^w on the primitive decomposition. Ψ^w is a faithful functor, since it is injective on both objects and labels.

2.6.4 Θ^f Again!

In the cases of framed, fenced and hedged bisimulation we have that two environments are $\mathcal{M}$-equal iff they are equal, so all m-arrows will disappear under $\equiv_u$. However, for substitutions we will see that "up to $\mathcal{M}$-equivalence" buys us something.

We let $\mathfrak{A}^m$ be $\mathfrak{A}$ up to $\mathcal{M}$-equivalence and $\equiv_u$, i.e., the category obtained from $\mathfrak{A}$ by first adding all arrows of the type $((\sigma,\rho),P,Q) \xrightarrow{m} ((\sigma',\rho'),P,Q)$ where $(\sigma,\rho) \gtrcurlyeq (\sigma',\rho')$, then closing the set of arrows under composition and finally taking the quotient with $\equiv_u$ as defined in Definition 2.6.4. The effect of this is to make $\mathcal{M}$–equivalent environments isomorphic.

We define $\Phi : \mathfrak{A}^m \to \mathfrak{H}$ as $\Phi((\sigma,\rho),P,Q) := (\varphi(\sigma,\rho),P,Q)$ and $\Phi(((\sigma,\rho),P,Q) \xrightarrow{\tilde{\mu}} ((\sigma',\rho'),P',Q')) := (\varphi(\sigma,\rho),P,Q) \xrightarrow{\tilde{\mu}'} (\varphi(\sigma',\rho'),P',Q')$. where $\tilde{\mu}' = \tilde{\mu} \setminus m$. The removal of the m–actions is valid since $\varphi(\sigma,\rho) = \varphi(\sigma',\rho')$ if and only if $(\sigma,\rho) \gtrcurlyeq (\sigma',\rho')$ according to Lemma 2.4.50. As in the proof of Theorem 2.5.21 we have that Φ is a functor.

We also define $\Theta^f : \mathfrak{H} \to \mathfrak{A}^m$ by letting $\Theta^f(h,P,Q) := (\theta^f(h),P,Q)$. The action of Θ^f on the primitive $\mathfrak{H}$-arrows is as follows:

1. If $(h,P,Q) \xrightarrow{\tau} (h',P',Q')$ then we define $\Theta^f(\xrightarrow{\tau}) := \xrightarrow{\tau}$.

2. If $(h,P,Q) \xrightarrow{a\,M} (h',P',Q')$ we let $B = \pi_1(h' \backslash h)$. We have that $\Theta^f(h,P,Q) \xrightarrow{a\,M} ((h_1^f\{^B/_Y\}, h_2^f\{^B/_Y\}), P', Q')$ by the proof of Theorem 2.5.21. By Lemma 2.4.44 there is an arrow $((h_1^f\{^B/_Y\}, h_2^f\{^B/_Y\}), P', Q') \xrightarrow{m} (\theta^f(h'), P', Q')$. We then define $\Theta^f(\xrightarrow{a\,M}) := \xrightarrow{m} \circ \xrightarrow{a\,M}$.
3. If $(h,P,Q) \xrightarrow{\overline{a}\,M} (h',P',Q')$ then we let b and N be the messages corresponding to a respective M according to h'. As in the proof of Theorem 2.5.22 we have that $\Theta^f(h,P,Q) \xrightarrow{\overline{a}\,M} ((h_1^f\{^M/_x\}, h_2^f\{^N/_x\}), P', Q')$. By Lemma 2.4.42 we get $\varphi(h_1^f\{^M/_x\}, h_2^f\{^N/_x\}) = h'$, so there is an arrow $((h_1^f\{^M/_x\}, h_2^f\{^N/_x\}), P', Q') \xrightarrow{m} (\theta^f(h'), P', Q')$. We then define $\Theta^f(\xrightarrow{\overline{a}\,M}) := \xrightarrow{m} \circ \xrightarrow{\overline{a}\,M}$.

By the above definition, it is clear that Θ^f is a functor.

Theorem 2.6.5 *$\mathfrak{A}^m$ and $\mathfrak{H}$ are equivalent. More precisely, $\Phi \circ \Theta^f = \mathrm{Id}_{\mathfrak{H}}$ and $\Theta^f \circ \Phi$ is isomorphic to $\mathrm{Id}_{\mathfrak{A}^m}$.*

Proof. By Lemma 2.4.39 we have that $h = \varphi(h_1^f, h_2^f)$ whenever h is consistent, so $(\Phi \circ \Theta^f)(h,P,Q) = (h,P,Q)$. Since Φ throws away all m-arrows introduced by Θ^f we have that $\Phi \circ \Theta^f = \mathrm{Id}_{\mathfrak{H}}$.
To show that $\Theta^f \circ \Phi$ is isomorphic to $\mathrm{Id}_{\mathfrak{A}^m}$ we need to find $\mathfrak{A}^m$-isomorphisms to complete the "naturality square" below. We use the m-arrows $((\sigma,\rho),P,Q) \xrightarrow{m} ((\theta_1^f(\varphi(\sigma,\rho)), \theta_2^f(\varphi(\sigma,\rho))), P, Q)$. There are such arrows according to Lemma 2.4.39 and they are isomorphisms since we only identify $\equiv_u$-equivalent arrows and $\mathcal{M}$-equivalence is symmetric. To exhibit the functor isomorphism we need to show that the following diagram commutes whenever $((\sigma,\rho),P,Q) \xrightarrow{\tilde{\mu}} ((\sigma',\rho'),P',Q')$.

$$\begin{array}{ccc} ((\sigma,\rho),P,Q) & \xrightarrow{m} & \Theta^f \circ \Phi((\sigma,\rho),P,Q) \\ \tilde{\mu}\downarrow & & \downarrow{\scriptstyle \Theta^f \circ \Phi(\tilde{\mu})} \\ ((\sigma',\rho'),P',Q') & \xrightarrow{m} & \Theta^f \circ \Phi((\sigma',\rho'),P',Q') \end{array}$$

Consider the arrows $((\sigma,\rho),P,Q) \xrightarrow{\tilde{\mu}\,m} \Theta^f \circ \Phi((\sigma',\rho'),P',Q')$ and $((\sigma,\rho),P,Q) \xrightarrow{m\,\Theta^f \circ \Phi(\tilde{\mu})} \Theta^f \circ \Phi((\sigma',\rho'),P',Q')$. As Θ^f and Φ only change labels by inserting and removing m-actions we have that $\xrightarrow{\tilde{\mu}\,m} \equiv_u \xrightarrow{m\,\Theta^f \circ \Phi(\tilde{\mu})}$. As the arrows in $\mathfrak{A}^m$ are equivalence classes with respect to $\equiv_u$ the diagram is commutative. □

In other words, $\mathfrak{H}$ is equivalent to "$\mathfrak{A}$ up to $\mathcal{M}$-equivalence and $\equiv_u$", where the isomorphism on the alley side is the normalization of environments given by $(\sigma,\rho) \mapsto (\theta_1^f(\varphi(\sigma,\rho)), \theta_2^f(\varphi(\sigma,\rho)))$.

We can characterize the relationship between $\mathfrak{F}_\#$ (fenced) and $\mathfrak{S}$ (trellis) in the very same way, namely that $\mathfrak{F}_\#$ is equivalent to "$\mathfrak{S}$ up to $\mathcal{M}$-equivalence and $\equiv_u$" ($\mathfrak{S}^m$), where the isomorphism on the alley side is the same normalization as above.

2.6.5 Summary

The relations between the different categories are graphically presented in Figure 2.2. With these results, we have related the bisimulations not only based on the distinguishing power of the environments, but also based on their internal structure.

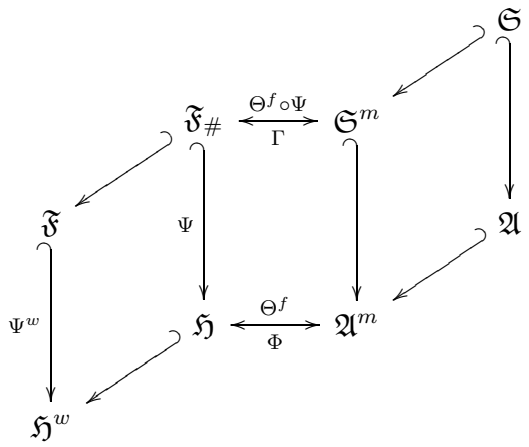

Figure 2.2: Categorical Relations

2.7 Conclusions

As an interpretation of the results of Section 2.2, we may underline two different deficiencies in the original definition of framed bisimulation. In a sense, it is at the same time both too weak and too strong with respect to barbed equivalence, for orthogonal reasons.

1. The definition is too weak in the sense that its authors did not impose a minimality requirement on the environment and argue that this "results in simpler definitions, and does not compromise soundness (w.r.t. testing equivalence)". However, when adding the minimality requirement, as done in fenced bisimulation, the relation becomes strictly stronger than barbed equivalence. As seen in the example in Section 2.2.1, it is not obvious how to choose the non-minimal extension on process output in order to get a bisimilar framed process pair. Thus, for example the purpose of mechanization, we regard fenced bisimilarity as better suited than framed, but both bisimilarities suffer from the second problem of being too strong.

2. The definition is too rigid, because it requires the syntactic coincidence of names received by the environment from the two processes under observation: whereas framed bisimulation requires identity, hedged bisimulation allows the environment to simply record that the names respectively received from the processes in a bisimulation step correspond (c.f. Section 2.2.3). This is the main reason why fenced bisimilarity does not coincide with barbed equivalence, while hedged bisimilarity does.

In Section 2.3, we developed a uniform framework for comparing environment-sensitive bisimilarities based on their environments, because the standard merely set-theoretic framework turned out not to be sufficiently general. We then used this new framework to reformulate and refine the results of the two previous comparisons of environment-sensitive bisimilarities by [EHHO99] and [FHJ01]. We found that [FHJ01] showed only the full abstraction of $\sim_{\#}$ with respect to $\sim_{\mathrm{s}}$, but not the other direction. Once this was clear, the missing full abstraction was easy to construct (c.f. Lemma 2.5.6).

We exercised the comparison framework in Section 2.5. In particular, we found that there exists a "framed-style" environment-sensitive bisimilarity, namely hedged, that is equivalent to barbed equivalence. We then used the distinguishing examples of Section 2.2 to show that we have fully characterized all full abstractions and equivalences between the bisimilarities.

Furthermore, in Section 2.6 we proposed a novel method for comparing environment-sensitive bisimilarities as categories. This technique allows us to study the internal structures of the bisimilarities. In particular, it allows us to highlight the fact that the non-minimality of framed bisimilarity distinguishes it conceptually from the other bisimilarities. After redefining some up-to techniques in this setting, we are at least able to embed framed bisimilarity into "hedged bisimilarity up to weakening". We also show that hedged bisimilarity is categorically equivalent to "alley bisimilarity up to $\mathcal{M}$-equivalence", thus refining the merely set-theoretic bisimilarity equivalence proved in Section 2.5.

The spi-calculus defined in Section 1.4 has a very simple expression language. However, our results are easy to adopt to a calculus with pairing, and we have found no reason to expect that compound keys or public-key cryptography would be more difficult to treat. We chose the more restricted setting only in order to focus the attention on the various bisimilarities. Moreover, although we only treat strong late bisimulations, the results carry over directly to the early and/or weak versions.

From a rather subjective point of view, when working with concrete examples we find hedged bisimulation easier to work with than alley bisimulation, because the knowledge of the environment in each reachable configuration is arguably easier to understand, and the verification of consistency is more straightforward. Moreover, the simpler structure of hedges allows a less complex proof of soundness up to

weakening than alley bisimilarity. On the other hand, alley bisimilarity benefits from an attractive logical characterization of environment consistency.

Chapter 3

Extending the Message Language

The spi calculus defined in Section 1.4 works with a minimal message language, only containing symmetric encryption with atomic keys. In order to model a cryptographic protocol we clearly need to extend this language, at the very least with tuples.

The formal cryptography tradition [DY83] is moving towards a more complete treatment of algebraic properties of cryptographic primitives [CDL06]. Examples also include a more fine-grained treatment of "compound primitives" such as block encryption algorithms used in electronic code book or cipher block chaining mode, and message authentication codes [KR04]. However, algorithms for these complex message algebras are often defined ad-hoc [CKRT03] and/or without termination guarantees (e.g., as ad-hoc additions to ProVerif [Bla01]). Recent work [BB05, AC06, Bau07] aims at finding a usefully large class of message algebras with a uniform decision procedure for secrecy properties. There are two main ways of specifying secrecy for a cryptographic protocol [CRZ07].

(1) One common approach is to see if the attacker can deduce the value of a secret parameter of the protocol, after some interaction with the protocol participants. This *disclosure*-based approach is taken in, e.g., [Low96, Sch96, KMM94].
(2) The other approach is to check whether the attacker can notice any difference between protocol runs with different values of the secret parameter. This *indistinguishability*-based approach is commonly used in the spi calculus.

In Section 3.1, we formally define disclosure and indistinguishability as state properties, i.e., at some point during a protocol run. In Section 3.2 we then prove that there exist message algebras in which disclosure is decidable but indistinguishability is not. The proof is by reducing the ambiguity problem for context-free grammars to an indistinguishability problem. Abadi and Cortier [AC04a, AC04b] gave a proof sketch for this separation result, based on another undecidable problem relating two

pairs of Turing machines. The work in Section 3.2 is, to the knowledge of the author, the first full proof.

In Section 3.3, we then define a class of *constructor-destructor* expression languages. This class is a subclass of the *subterm-convergent* expression languages [AC06], with similarities to the *data term* languages used by Baldamus et al. [BPV05]. The virtue of our class of languages (compared to both of the above) is that it allows a smooth generalization of the notions of analysis, synthesis and irreducibles (Definition 3.3.4, Lemma 3.3.12), and enables a symbolic operational semantics that is faithful with respect to scope extrusions (Theorem 4.1.18). We validate our definitions by showing that consistency for hedges over one of these message languages corresponds to static equivalence over a closely related language (Theorem 3.3.24). In relation to the process language, the two sides of an irreducible hedge can be distinguished by some guard formula iff the hedge is inconsistent (Theorem 3.3.25).

Finally, in Section 3.4 we redefine the spi calculus for any class of expression languages that admit a reasonable notion of evaluation. This notably includes the constructor-destructor and subterm-convergent expression languages.

3.1 Message Algebras

Definition 3.1.1 *A signature* $\Sigma = (\mathcal{F}, \mathrm{ar})$ *over* $\mathcal{N}$ *with variables from* $\mathcal{V}$ *is a finite set of function symbols* $\mathcal{F} \ni \mathtt{f}$*, and a function* $\mathrm{ar} : \mathcal{F} \to \mathbb{N}$ *giving their arity (which may be 0). The set of terms* $\mathcal{T}_\Sigma$ *is then defined by* $t, u ::= a \mid x \mid \mathtt{f}(t_1, \ldots, t_n)$ *where* $\mathrm{ar}(\mathtt{f}) = n$*. Let* $|t|_u$ *be the number of occurrences of* u *in* t*. We let* $\mathrm{n}(t)$ *be the set of names and* $\mathrm{v}(t)$ *the set of variables occurring in a term* t*. The concrete terms* $\mathcal{T}_\Sigma^c$ *are those that do not contain any variables.*

In algebras for cryptography, message equality is typically induced by some set of equations. In the case of symmetric cryptography, this may be as simple as the single rule $\mathrm{D}_k(\mathrm{E}_k(x)) = x$ previously seen, stating that a message x encrypted under the key k can be decrypted using the same key. If this equation is oriented from left to right we obtain a *rewrite rule*, where the resulting rewrite system is *subterm convergent* (i.e., the rewriting of any term converges to some subterm of the initial term).

In order to add more operations and more accurately model the behavior of particular implementations of cryptographic primitives, one can simply add to and modify the rule set. One drawback with such refinements is that the rewrite system might no longer be convergent (e.g., when adding associative and commutative operators like `xor`), so even the decidability of equality must be proven for each additional operation. (However, composition of disjoint message algebras preserves decidability [ACD07].)

Since we make a distinction between channel names and non-name expressions in the semantics of the spi calculus, we correspondingly permit rewrite rules that apply only to names.

Definition 3.1.2 *A rewrite rule is of the form "$t_1 \to t_2$ if ϕ", where $t_1, t_2 \in \mathcal{T}_\Sigma$ and ϕ is a conjunction of membership predicates $x_i \in \mathcal{N}$. We require* $\mathrm{v}(t_2) \cup \mathrm{v}(\phi) \subseteq \mathrm{v}(t_1)$ *and* $\mathrm{n}(t_1) = \mathrm{n}(t_2) = \emptyset$. *A term t matches a rewrite rule of the form above if there is a substitution $\sigma : \mathrm{v}(t_1) \to \mathcal{T}_\Sigma$ such that $t = t_1\sigma$ and $\phi\sigma$ is true. If E is a (finite) set of rewrite rules containing this rule, t can be* head rewritten *to $t_2\sigma$ under E, which we write $t \to^{\mathrm{H}}_E t_2\sigma$. We let $\to_E$ be the closure of $\to^{\mathrm{H}}_E$ under contexts, $\to^*_E$ be the transitive and reflexive closure of $\to_E$, and $\equiv_E$ be the transitive, reflexive and symmetric closure of $\to_E$. When E is clear from the context, we often omit it.*

In what follows, we will assume that $\equiv_E$ is decidable; this is notably the case if the rewrite system $\to_E$ is confluent and terminating. For these (*convergent*) rewrite systems, we write $t{\downarrow_E}$ for the unique term such that $t \to^*_E t{\downarrow} \not\to_E$.

3.1.1 Frames and Operations

The most important dynamic characteristic of a Dolev-Yao adversary is the set of messages that it has learned by communicating with the legitimate participants of the protocol. This message set is the only information needed to verify if the adversary/environment knows a particular (confidential) datum. For the indistinguishability-based approach we want to compare results of corresponding operations on the knowledge of two adversaries, so we need some way of relating the corresponding messages. One way of doing this, used for alley bisimulation [BDP02], is to represent the knowledge as a substitution (called a *knowledge environment*). Here, messages known to two different adversaries (i.e., in the range of the corresponding substitutions) are related if they have the same pre-image.

As usual, the adversary can apply any combination of cryptographic functions to the messages it possesses. It can also generate names subject to a freshness constraint. In [AC06], this constraint is made explicit, in that the adversary knowledge is augmented with a set of names that may not be freshly generated. For simplicity and continuity with the preceding chapter, we instead follow Boreale et al. [BDP02] in requiring that whenever an environment constructs a message, the freshly generated names (nonces and keys) must be different from all the names in the environment ($\mathrm{n}(\mathrm{range}(\sigma))$).

Definition 3.1.3 *The knowledge environment σ can* primitively generate *the message (term) t, written $\sigma \vdash^p t$, if there is t' such that $\mathrm{n}(t') \cap \mathrm{n}(\mathrm{range}(\sigma)) = \emptyset$, $\mathrm{v}(t') \subseteq \mathrm{dom}(\sigma)$ and $t'\sigma = t$. Given an equational theory E, the environment σ* generates *t in E, written $\sigma \vdash_E t$, if there is u such that $\sigma \vdash^p u$ and $u \equiv_E t$.*

Two environments σ_1 *and* σ_2 *where* $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2)$ *are* indistinguishable *under* E*, written* $\sigma_1 \cong_E \sigma_2$*, if for all* t, u *with* $\mathrm{n}(t,u) \cap \mathrm{n}(\mathrm{range}(\sigma_1) \cup \mathrm{range}(\sigma_2)) = \emptyset$ *and* $(\mathrm{v}(t) \cup \mathrm{v}(u)) \subseteq \mathrm{dom}(\sigma_1)$*, we have* $t\sigma_1 \equiv_E u\sigma_1$ *iff* $t\sigma_2 \equiv_E u\sigma_2$*.*

In regard to automated verification, since $\mathcal{T}_\Sigma$ is enumerable we immediately get that the message construction problem is semidecidable and the indistinguishability problem is co-semidecidable (by enumerating all t, u and checking the conditions) whenever $\equiv_E$ is decidable. An important question for automated verification is finding message algebras in which these problems are (efficiently) decidable. In [AC06], the authors proved that in message algebras containing a function symbol not present in rewrite rules (e.g., hashing), decidability of $\cong_E$ implies decidability of $\vdash_E$. Moreover, they gave an example of a convergent rewrite system E_{AC}, and gave a proof sketch for the result that $\vdash_{E_{\mathrm{AC}}}$ was decidable but $\cong_{E_{\mathrm{AC}}}$ undecidable. We now exhibit another rewrite system with the same properties but in a simpler setting (context-free grammars versus Turing machines), and develop a full proof.

3.2 Static Equivalence *is* Harder than Knowledge

Our example message algebra, where deduction is decidable but static equivalence is not, is based on leftmost derivations of context-free grammars in Chomsky normal form. We first recall some definitions for such grammars.

A context-free grammar $G = (A_G, X_G, s_G, T_G \cup N_G)$ in Chomsky normal form (CNF) consists of terminal symbols A_G, non-terminal symbols X_G (with $A_G \cap X_G = \emptyset$), an initial symbol $s_G \in X_G$, and two kinds of derivation rules: *terminal* and *non-terminal* rules. Terminal rules $(n \to t) \in T_G$ take a non-terminal symbol n to a terminal symbol t, whereas non-terminal rules $(n \to n_1 n_2) \in N_G$ take a non-terminal symbol to two non-terminal symbols.

A leftmost derivation of $\widetilde{w} \in A_G^* X_G^*$ is a word $r_1 \cdots r_k \in (T_G \cup N_G)^*$ where there exist words $\widetilde{a}^0, \widetilde{a}^1, \ldots, \widetilde{a}^k \in A_G^*$ and $\widetilde{x}^0, \widetilde{x}^1, \ldots, \widetilde{x}^k \in X_G^*$ such that $\widetilde{a}^0 \widetilde{x}^0 = s_G$, $\widetilde{a}^k \widetilde{x}^k = \widetilde{w}$ and for all $i = 1, \ldots, k$ we have that either $r_i = (n \to t) \in T_G$, $\widetilde{a}^i = \widetilde{a}^{i-1} t$ and $n\widetilde{x}^i = \widetilde{x}^{i-1}$, or $r_i = (n \to n_1 n_2) \in N_G$, $\widetilde{a}^i = \widetilde{a}^{i-1}$ and $\widetilde{x}^{i-1} = n\widetilde{y}$ and $\widetilde{x}^i = n_1 n_2 \widetilde{y}$ for some $\widetilde{y}$. It is easy to show that k above (the length of the derivation) is equal to $|\widetilde{w}| + |\widetilde{a}^n| - 1$. Such a derivation is called partial if $\widetilde{w} \notin A_G^*$. The language of a grammar $L(G)$ is the set of words over A_G that have a leftmost derivation.

A grammar in CNF has no dead-end non-terminals, in the sense that $\forall x \in X_G \ \exists \widetilde{w} \in L(A_G, X_G, x, T_G \cup N_G)$. Moreover, all non-terminals are reachable from the initial symbol, meaning that $\forall x \in X_G \ \exists \widetilde{w}_1, \widetilde{w}_2, \widetilde{r}$ such that $\widetilde{r}$ is a leftmost G-derivation of $\widetilde{w}_1 x \widetilde{w}_2$

A grammar G is *ambiguous* if there exists a word $\widetilde{w} \in L(G)$ that has two different leftmost derivations. A classical result in formal language theory is the undecidability of whether a given context-free grammar (in CNF) is ambiguous. In what follows,

we define a rewrite system such that a grammar G is ambiguous iff a pair of knowledge environments (depending on G) are indistinguishable under the equivalence induced by the rewrite system.

Example 3.2.1 *As a running example, let us consider a context-free grammar for a parenthesis language. Let* $G_{\mathrm{p}} := (\{\mathtt{l}, \mathtt{r}, \mathtt{a}\}, \{\mathtt{S}, \mathtt{S}', \mathtt{L}, \mathtt{R}\}, \mathtt{S}, T_G \cup N_G)$ *where* $T_G := \{\mathtt{S} \to \mathtt{a}, \mathtt{L} \to \mathtt{l}, \mathtt{R} \to \mathtt{r}\}$ *and* $N_G := \{\mathtt{S} \to \mathtt{SS}, \mathtt{S} \to \mathtt{LS}', \mathtt{S}' \to \mathtt{SR}\}$. *It is straightforward to verify that* G_{p} *is in CNF. GP is ambiguous, since there are two different leftmost derivations of the word* `aaa`*:*

$$\mathtt{S} \to \mathtt{SS} \to \mathtt{aS} \to \mathtt{aSS} \to \mathtt{aaS} \to \mathtt{aaa}$$
$$\mathtt{S} \to \mathtt{SS} \to \mathtt{SSS} \to \mathtt{aSS} \to \mathtt{aaS} \to \mathtt{aaa}$$

3.2.1 Message Algebra

We introduce a message algebra intended to model given leftmost derivations according to the rules of context-free grammars in Chomsky normal form. Let Σ be the following signature.

Symbol	Arity	Intuitive meaning
`Nil`	0	Nil
`id`	1	Non-terminal identifier
$(\cdot\,.\,\cdot)$	2	Pair
`OK`	2	Name type check
`T`	2	Terminal grammar rule
`N`	3	Non-terminal grammar rule
`dc`	5	Derivation context

Definition 3.2.2 *We define the following shorthand notations for terms.*

lists *Let* $[\epsilon] := \mathtt{Nil}$ *and* $[\widetilde{w}v] := (v\ .\ [\widetilde{w}])$.
grammar rules *Let* $\mathrm{rule}(k \to lm) := \mathtt{N}(k, l, m)$ *and* $\mathrm{rule}(n \to a) := \mathtt{T}(n, a)$.
derivations *Let* $\mathrm{der}_x(\epsilon) := x$ *and* $\mathrm{der}_x(r_1\widetilde{s}) := (\mathrm{rule}(r_1)\ .\ \mathrm{der}_x(\widetilde{s}))$.
derivation lengths *Let* $dl(0) := \mathtt{Nil}$ *and* $dl(n+1) := (\mathtt{OK}(\mathtt{Nil}, \mathtt{Nil})\ .\ dl(n))$.

The five arguments of the derivation context (`dc`) have the following meanings:

1 The symbol with which a derivation started.
2 Ensures that rewriting does not reduce the size of terms, and counts the number of rules that have been applied. (After k rule applications, this argument should contain the term $dl(k)$.)
3 A list of terminals forming a prefix of the word that is derived.

4 A list of the non-terminal suffix that remains to be rewritten.
5 A list of the derivation rules to apply in order.

Let E_{CNF} be the equational theory on Σ induced by the following rewrite rules:

$$\mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ (\mathtt{T}(y,t)\ .\ u)) \rightarrow \mathtt{dc}(y,\ (\mathtt{OK}(\mathtt{Nil},\mathtt{Nil})\ .\ \mathtt{Nil}),\ (t\ .\ \mathtt{Nil}),\ \mathtt{Nil},\ u) \quad (3.1)$$

$$\mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ (\mathtt{N}(y,t_1,t_2)\ .\ u)) \rightarrow \mathtt{dc}(y,\ (\mathtt{OK}(\mathtt{Nil},\mathtt{Nil})\ .\ \mathtt{Nil}),\ \mathtt{Nil},\ (t_1\ .\ (t_2\ .\ \mathtt{Nil})),\ u) \quad (3.2)$$

$$\mathtt{dc}(v,\ w,\ x,\ (y\ .\ z),\ (\mathtt{T}(y,t)\ .\ u)) \rightarrow \mathtt{dc}(v,\ (\mathtt{OK}(y,y)\ .\ w),\ (t\ .\ x),\ z,\ u) \quad (3.3)$$

$$\mathtt{dc}(v,\ w,\ x,\ (y\ .\ z),\ (\mathtt{N}(y,t_1,t_2)\ .\ u)) \rightarrow \mathtt{dc}(v,\ (\mathtt{OK}(y,y)\ .\ w),\ x,\ (t_1\ .\ (t_2\ .\ z)),\ u) \quad (3.4)$$

$$\mathtt{OK}(m,n) \rightarrow \mathtt{OK}(\mathtt{Nil},\mathtt{Nil}) \text{ when } m,n \in \mathcal{N} \quad (3.5)$$

Note that these rules are terminating and confluent when oriented left to right, so the equality problem is clearly decidable. Intuitively, the rules denote the following operations related to leftmost derivations:

(1) Initial derivation step, using a terminal rule.
(2) Initial derivation step, using a nonterminal rule.
(3) Subsequent derivation step, using a terminal rule.
(4) Subsequent derivation step, using a nonterminal rule.
(5) Hiding of the non-terminal that is discharged (iff it is a name). Here, the term on the right-hand side is chosen to be "inert" and of the same syntactic size as the left-hand side.

Theorem 3.2.3 *The construction problem for E_{CNF} is decidable.*

Proof. By inspection, the rewrite rules have the property that $T \rightarrow T'$ implies that $|T| \leq |T'|$, so no term is of greater syntactic size than its normal form. Thus, all equivalence classes are finite modulo injective renaming. To check deductibility, we check if any of a finite (modulo injective renaming as above) number of terms can be primitively generated, which clearly is decidable. □

3.2.2 Translation

Given the rewrite system above and a context-free grammar, we look for a pair of knowledge environments that are indistinguishable if and only if the grammar is unambiguous. The idea is that one environment (σ_G below) contains the actual grammar rules, whereas the other contains their names. The domains of the substitutions are chosen arbitrarily. The two environments should then be statically equivalent if and only if different leftmost derivations always yield different results.

Definition 3.2.4 *If $G := (A_G, X_G, s_G, T_G \cup N_G)$ is in CNF, $A_G \cup X_G \subset \mathcal{N}$ and $f_{\mathcal{X}} : \mathcal{N} \times \mathcal{N} \to \mathcal{X}$ and $g_{\mathcal{X}} : \mathcal{N} \times \mathcal{N} \times \mathcal{N} \to \mathcal{X}$ for $\mathcal{X} = \mathcal{V}, \mathcal{N}$ are injective functions with $\mathrm{range}(f_{\mathcal{X}}) \cap \mathrm{range}(g_{\mathcal{X}}) = \emptyset$, then we let*

$$\begin{aligned}
\sigma_G := \Big(&\Big\{ \big\{ {}^{\mathtt{T}(a,b)}/_{f_{\mathcal{V}}(a,b)} \big\} \mid (a \to b) \in T_G \Big\} \\
&\cup \Big\{ \big\{ {}^{\mathtt{N}(a,b,c)}/_{g_{\mathcal{V}}(a,b,c)} \big\} \mid (a \to bc) \in N_G \Big\} \Big), \\
\rho_G := \Big(&\Big\{ \big\{ {}^{\mathtt{id}(f_{\mathcal{N}}(a,b))}/_{f_{\mathcal{V}}(a,b)} \big\} \mid (a \to b) \in T_G \Big\} \\
&\cup \Big\{ \big\{ {}^{\mathtt{id}(g_{\mathcal{N}}(a,b,c))}/_{g_{\mathcal{V}}(a,b,c)} \big\} \mid (a \to bc) \in N_G \Big\} \Big)
\end{aligned}$$

Example 3.2.5 *For the grammar G_{p}, the corresponding environments are*

$$\begin{aligned}
\sigma_{G_{\mathrm{p}}} &= \big\{ {}^{\mathtt{T(S,a)}}/_{x_1} \big\} \big\{ {}^{\mathtt{T(L,l)}}/_{x_2} \big\} \big\{ {}^{\mathtt{T(R,r)}}/_{x_3} \big\} \big\{ {}^{\mathtt{N(S,S,S)}}/_{x_4} \big\} \big\{ {}^{\mathtt{N(S,L,S')}}/_{x_5} \big\} \big\{ {}^{\mathtt{N(S',S,R)}}/_{x_6} \big\} \\
\rho_{G_{\mathrm{p}}} &= \big\{ {}^{\mathtt{id}(a_1)}/_{x_1} \big\} \big\{ {}^{\mathtt{id}(a_2)}/_{x_2} \big\} \big\{ {}^{\mathtt{id}(a_3)}/_{x_3} \big\} \big\{ {}^{\mathtt{id}(a_4)}/_{x_4} \big\} \big\{ {}^{\mathtt{id}(a_5)}/_{x_5} \big\} \big\{ {}^{\mathtt{id}(a_6)}/_{x_6} \big\}
\end{aligned}$$

where $x_1, x_2, x_3, x_4, x_5, x_6$ ($a_1, a_2, a_3, a_4, a_5, a_6$) are pairwise different variables (names).

At the corresponding point in the proof of [AC04a] (Proposition 5, page 17) the authors conclude with: "Then we can verify that two machines $\mathcal{M}(M_1, M_2)$ and $\mathcal{M}(M_1', M_2')$ verify the [undecidable] property (P) [...] if and only if $\phi_{\mathcal{M}(M_1,M_2)} \cong \phi_{\mathcal{M}(M_1',M_2')}$." However, they say nothing of *how* to verify that. To clarify this for ourselves and others, we devote the remainder of this section to a proof of this proposition in our setting.

3.2.3 Derivations

In what follows, we assume a fixed context-free grammar G in CNF where $G := (A_G, X_G, s_G, T_G \cup N_G)$. The following lemma shows that partial derivations of G can be simulated by the rewrite system.

Lemma 3.2.6 *Let $\mathrm{tail}^k(\widetilde{w}) := w_{k+1} \dots w_{|w|}$. Then $s_G \to_G^k \widetilde{a}\widetilde{n}$ using the partial leftmost derivation $\widetilde{r} := r_1 r_2 \dots r_k$, where $\widetilde{a} \in A_G^*$ and $\widetilde{n} \in X_G^*$, iff for any term t,*

$$\mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathrm{der}_t(\widetilde{r})) \to^{2k-1} \mathtt{dc}(s_G,\ dl(k),\ [\widetilde{a}],\ [\widetilde{n}],\ t).$$

Proof. By induction on k. □

Example 3.2.7 *A leftmost derivation of the word* `lara` *by* G_{p} *is given by* $\widetilde{r} := \mathtt{S} \to \mathtt{SS}, \mathtt{S} \to \mathtt{LS}, \mathtt{L} \to \mathtt{L}, \mathtt{S}' \to \mathtt{SR}, \mathtt{S} \to \mathtt{a}, \mathtt{R} \to \mathtt{r}, \mathtt{S} \to \mathtt{a}$ *(i.e.,* $\mathtt{S} \to \mathtt{SS} \to \mathtt{LS'S} \to \mathtt{lS'S} \to \mathtt{lSRS} \to \mathtt{laRS} \to \mathtt{larS} \to \mathtt{lara}$*). Moreover,*

$$
\begin{aligned}
&\mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathrm{der}_{\mathtt{Nil}}(\widetilde{r}))\\
&\qquad = \mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ (\mathtt{N}(\mathtt{S},\mathtt{S},\mathtt{S})\ .\ \mathrm{der}_{\mathtt{Nil}}(\mathrm{tail}^1(\widetilde{r}))))\\
&\quad \to \mathtt{dc}(\mathtt{S},\ dl(1),\ \mathtt{Nil},\ (\mathtt{S}\ .\ (\mathtt{S}\ .\ \mathtt{Nil})),\ (\mathtt{N}(\mathtt{S},\mathtt{L},\mathtt{S}')\ .\ \mathrm{der}_{\mathtt{Nil}}(\mathrm{tail}^2(\widetilde{r}))))\\
&\to \mathtt{dc}(\mathtt{S},\ (\mathtt{OK}(\mathtt{S},\mathtt{S})\ .\ dl(1)),\ \mathtt{Nil},\ (\mathtt{L}\ .\ (\mathtt{S}'\ .\ (\mathtt{S}\ .\ \mathtt{Nil}))),\ \mathrm{der}_{\mathtt{Nil}}(\mathrm{tail}^2(\widetilde{r})))\\
&\to \mathtt{dc}(\mathtt{S},\ dl(2),\ \mathtt{Nil},\ (\mathtt{L}\ .\ (\mathtt{S}'\ .\ (\mathtt{S}\ .\ \mathtt{Nil}))),\ (\mathtt{T}(\mathtt{L},\mathtt{l})\ .\ \mathrm{der}_{\mathtt{Nil}}(\mathrm{tail}^3(\widetilde{r}))))\\
&\to \mathtt{dc}(\mathtt{S},\ (\mathtt{OK}(\mathtt{L},\mathtt{L})\ .\ dl(2)),\ (\mathtt{l}\ .\ \mathtt{Nil}),\ (\mathtt{S}'\ .\ (\mathtt{S}\ .\ \mathtt{Nil})),\ \mathrm{der}_{\mathtt{Nil}}(\mathrm{tail}^3(\widetilde{r})))\\
&\qquad \to \cdots \to \mathtt{dc}(\mathtt{S},\ dl(7),\ (\mathtt{a}\ .\ (\mathtt{r}\ .\ (\mathtt{a}\ .\ (\mathtt{l}\ .\ \mathtt{Nil})))),\ \mathtt{Nil},\ \mathtt{Nil}).
\end{aligned}
$$

Lemma 3.2.6 can be generalized to show that $\sigma_G \vdash_{E_{\mathrm{CNF}}}$ accurately models leftmost derivations of the grammar G.

Proposition 3.2.8 *If* $w \in A_G^*$ *then* $w \in L(G)$ *iff* $\sigma_G \vdash_{E_{\mathrm{CNF}}} \mathtt{dc}(s_G,\ dl(1+2|w|),\ [w],\ \mathtt{Nil},\ \mathtt{Nil})$.

Proof.

⇒ Assume that $w \in L(G)$. Then there exists a leftmost derivation $s_G \to^* w$ described by the tuple $\widetilde{r} := r_1 r_2 \ldots r_{2|w|-1}$. By Lemma 3.2.6 we have

$$\mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathrm{der}_{\mathtt{Nil}}(\widetilde{r})) \to^{4|w|-3} \mathtt{dc}(s_G,\ dl(1+2|w|),\ [w],\ \mathtt{Nil},\ \mathtt{Nil}).$$

Clearly $\sigma_G \vdash^p \mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathrm{der}_{\mathtt{Nil}}(\widetilde{r}))$.

⇐ Assume that $\sigma_G \vdash_{E_{\mathrm{CNF}}} U := \mathtt{dc}(s_G,\ dl(1+2|w|),\ [w],\ \mathtt{Nil},\ \mathtt{Nil})$. Then there exists $U' \equiv_{E_{\mathrm{CNF}}} U$ such that $\sigma_G \vdash^p U'$. Note that no rule creates a `dc` function symbol at the top level if there was not already one. Thus, since the environment does not contain any `dc` symbols, at the top level of U' there must be a `dc` function application.

By inspection of the grammar rules, no subterm of $[w]$ except for `Nil` is deducible. Thus, by inspection of the rewrite rules, the subterm $[w]$ of U must have been generated by repeated application of rule (3.1) or (3.3), consuming $\mathtt{T}(x,t)$ terms where $t \in A_G$.

Note that all terms in the environment σ_G are in normal form. Since no rewrite rule introduces a `T` function symbol, any $\mathtt{T}(x,t)$ where $t \in A_G$ are from range(σ_G), and thus $x \in X_G$.

In other words, whenever the third argument to the top-level `dc` function symbol grows (rules (3.1) and (3.3)), it is by using a terminal rule of G. Since the fourth argument only shrinks by application of rule (3.3), we can conclude that it always is a list of non-terminal symbols of the grammar.
By a similar argument, whenever the fourth argument to the top-level `dc` function symbol grows (rules (3.2) and (3.4)), it is by using a non-terminal rule of G. From this follows that there must exist $\widetilde{r}$ such that the last argument of the top-level `dc` function symbol of U' is equal to $\mathrm{der}_{\mathtt{Nil}}(\widetilde{r})$.
The subterm s_G of U is not fresh, so it is not deducible. By inspection of the rules, it must have been generated using rule (3.1) or (3.2). Thus, $U' = \mathtt{dc}(\mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, \mathrm{der}_{\mathtt{Nil}}(\widetilde{r}))$, so by Lemma 3.2.6 $s_G \rightarrow^* w$.

□

Our main technical lemma is a full characterization of the terms that can be derived by σ_G, in the case where G is unambiguous. When starting from a primitively generated term that was in normal form before applying the substitution, rewrite rules can only be applied as intended (derivation steps of the grammar G). To show this, we define and use a deterministic rewrite strategy.

Lemma 3.2.9 *Let G be fixed as above, and assume that G is unambiguous. Let $\mathcal{L}'_0$ be the set of (possibly open) terms in normal form that do not contain any name in $A_G \cup X_G$. Let $\mathcal{D}_0(x) := \{\mathtt{dc}(\mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, x)\}$ and for $k > 0$*

$$\mathcal{D}_k(x) := \{\mathtt{dc}(n,\ dl(k),\ [\widetilde{a}],\ [\widetilde{n}],\ x) \mid \widetilde{a} \in A_G^* \wedge \widetilde{n} \in X_G^* \wedge n \rightarrow_G^k \widetilde{a}\widetilde{n} \text{ using a leftmost partial derivation}\}$$

Let the sets $\mathcal{L}'_k$ for $k > 0$ be the smallest sets satisfying the rule (DER) *below.*

$$(\text{DER})\ \frac{U \in \mathcal{L}'_k}{U\{^W/_x\} \in \mathcal{L}'_{k+l\cdot|U|_x}}\ \text{IF } k \geq 0, l > 0 \text{ AND } \exists V \in \mathcal{L}'_0 \text{ WITH } W \in \mathcal{D}_l(V)$$

Let $\mathcal{L}_k := \{U\sigma_G \mid U \in \mathcal{L}'_k \wedge \mathrm{v}(U) \subseteq \mathrm{dom}(\sigma_G)\}$ and $\mathcal{L} := \cup_{k\in\mathbb{N}}\mathcal{L}_k$. Note that the $\mathcal{L}_k$ are disjoint for different k. We then have:

1. *If $\sigma_G \vdash_{E_{\mathrm{CNF}}} U$, then $U{\downarrow} \in \mathcal{L}$.*
2. *If $U, U' \in \mathcal{L}_0$ and $U \equiv_{E_{\mathrm{CNF}}} U'$, then $U = U'$.*

Proof. Assume a well-ordering on contexts compatible with the partial well-ordering induced by the depth of the hole, and let $\rightsquigarrow$ be rewriting where the redex with the greatest context is always chosen. Note that this strategy is deterministic and complete (meaning that if $t \rightarrow$ then $t \rightsquigarrow$).

Let $P(i)$ be the conjunction of (I) and (II) below:

(I) If $U_0 \in \mathcal{L}_0$ and $U_0 \rightsquigarrow^* U_i \in \mathcal{L}_i$ where $U_i \rightsquigarrow$ then one of (a) to (d) holds.

(a) $U_i \rightsquigarrow_{(1)} U_{i+1} \in \mathcal{L}_{i+1}$ by some $\mathcal{D}_0((\mathtt{T}(y,t)\ .\ u)) \ni \overline{U} \rightarrow^{\mathrm{H}}_{(1)} \in \mathcal{D}_1(u)$ where $\mathtt{T}(y,t) \in \mathrm{range}(\sigma_G)$; or

(b) $U_i \rightsquigarrow_{(2)} U_{i+1} \in \mathcal{L}_{i+1}$ by some $\mathcal{D}_0(\mathtt{N}(y,t_1,t_2)) \ni \overline{U} \rightarrow^{\mathrm{H}}_{(2)} \in \mathcal{D}_1(u)$ where $\mathtt{N}(y,t_1,t_2) \in \mathrm{range}(\sigma_G)$; or

(c) $U_i \rightsquigarrow_{(3)} U_{i.5} \rightsquigarrow_{(5)} U_{i+1} \in \mathcal{L}_{i+1}$ by some $\mathcal{D}_j((\mathtt{T}(y,t)\ .\ u)) \ni \overline{U} \rightarrow^{\mathrm{H}}_{(3)} \rightarrow \in \mathcal{D}_{j+1}(u)$ where $\mathtt{T}(y,t) \in \mathrm{range}(\sigma_G)$; or

(d) $U_i \rightsquigarrow_{(4)} U_{i.5} \rightsquigarrow_{(5)} U_{i+1} \in \mathcal{L}_{i+1}$ by some $\mathcal{D}_j((\mathtt{N}(y,t_1,t_2)\ .\ u)) \ni \overline{U} \rightarrow^{\mathrm{H}}_{(4)} \rightarrow \in \mathcal{D}_{j+1}(u)$ where $\mathtt{N}(y,t_1,t_2) \in \mathrm{range}(\sigma_G)$.

(II) For each $U'_0 \in \mathcal{L}_0$ such that $U'_0 \rightsquigarrow^* U'_i \in \mathcal{L}_i$ and $U_i \rightsquigarrow^* U'_{i+1} \in \mathcal{L}_{i+1}$ as above, we have that $U'_{i+1} = U_{i+1}$ implies $U'_0 = U_0$.

We now show that $P(i)$ holds for all $i \in \mathbb{N}$, by induction on i.

- Base case: $i = 0$; we seek to show $P(0)$. Take $U_0 \in \mathcal{L}_0$, and let $U \in \mathcal{L}'_0$ be such that $U_0 = U\sigma_G$. Let $\overline{U_0}$ be the redex of U_0 with the greatest context C_0, such that $U_0 = C_0[\overline{U_0}]$ and $\overline{U_0} \rightarrow^{\mathrm{H}} V$.

  (I) Since U is in normal form, $\mathrm{range}(\varphi_G) \cap \mathcal{N} = \emptyset$ and $\mathrm{range}(\varphi_G)$ does not contain $\mathtt{OK}$ symbols, we have that $\overline{U_0} \not\rightarrow^{\mathrm{H}}_{(3.5)}$. We show that $\overline{U_0} \not\rightarrow^{\mathrm{H}}_{(3,4)}$ by contradiction.

  – Assume that $\overline{U_0} = \mathtt{dc}(v,\ w,\ x,\ (y\ .\ z),\ (t\ .\ u))$ where $t = \mathtt{N}(y,t_1,t_2)$ or $t = \mathtt{T}(y,t_1)$ for some x,y,z,t_1,t_2,u,v,w.
  If $t \notin \mathrm{range}(\sigma_G)$, then $U = C[\mathtt{dc}(v',\ w',\ x',\ (y'\ .\ z'),\ (t'\ .\ u'))]$ where $t' = \mathtt{N}(y',t'_1,t'_2)$ or $t' = \mathtt{T}(y',t'_1)$ for some $C,x',y',z',\ t'_1,t'_2,u',v',w'$, by the injectivity of σ_G and since $\mathrm{range}(\sigma_G)$ does not contain $\mathtt{dc}$ or $(\cdot\ .\ \cdot)$ symbols. Thus $U \rightarrow$, which is a contradiction.
  If $t \in \mathrm{range}(\sigma_G)$ then y can not be generated due to freshness constraints. By inspection of $\mathrm{range}(\sigma_G)$ we can only generate y inside a $\mathtt{T}$ or $\mathtt{N}$, which contradicts the assumption on the structure of $\overline{U_0}$.

  We may then assume that $\overline{U_0} = \mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ (x\ .\ u))$ where $x = \mathtt{T}(y,t)$ or $x = \mathtt{N}(y,t_1,t_2)$. Clearly $\overline{U_0} \in \mathcal{D}_0((x\ .\ u))$. As above, if $x \notin \mathrm{range}(\sigma_G)$ then $U \rightarrow$, which is a contradiction. We then have $\overline{U_0} \rightarrow^{\mathrm{H}} \in \mathcal{D}_1(u)$, so $U_0 \rightsquigarrow U_1 \in \mathcal{L}_1$.

  (II) Take $U'_0 \in \mathcal{L}_0$ such that $U'_0 \rightsquigarrow^* U'_1 \in \mathcal{L}_1$ where $U'_1 = U_1$. Let $\overline{U'_0}$ be the redex of U_0 with the greatest context C'_0, such that $U'_0 = C'_0[\overline{U'_0}]$ and $\overline{U'_0} \rightarrow^{\mathrm{H}} V'$. By (I) above, $\overline{U'_0} \not\rightarrow^{\mathrm{H}}_{(3,4,5)}$ and $V' \in \mathcal{D}_1(\mathcal{T}_\Sigma)$. Since V (resp. V') is the only subterm of U_1 (resp U'_1) in $\mathcal{D}_1(\mathcal{T}_\Sigma)$, we must have $C_0 = C'_0$ and $V = V'$. Since the rules (1) and (2) are injective, we have $\overline{U_0} = \overline{U'_0}$. Thus $U_0 = U'_0$.

- Induction case: Assume that $U_0 \in \mathcal{L}_0$ and $U_0 \rightsquigarrow^* U_i \in \mathcal{L}_i$ where $U_i \rightsquigarrow$. Moreover, let $U \in \mathcal{L}'_0$ be such that $U_0 = U\sigma_G$. Let $\overline{U_i}$ be the redex of U_i with the greatest context C_i, such that $U_i = C_i[\overline{U_i}]$ and $\overline{U_i} \to^{\mathrm{H}}$.
  To compare terms in different stages of $\rightsquigarrow$-rewriting, we let $\cong_C$ for a context C relate terms (or contexts) that coincide down to (exclusive) the depth of the "hole" in C and on the content (or position) of the "hole".

  (I) Let $U \in \mathcal{L}'_0$ be such that $U_0 = U\sigma_G$. By the properties of $\rightsquigarrow$, $U_0 = C_0[W]$ for some $W \rightsquigarrow^* \overline{U_i}$ and $C_0 \cong_{C_0} C_i$. There are five possibilities for $\overline{U_i} \to^h$.

  1 If $\overline{U_i} \to^{\mathrm{H}}_{(1)}$, then $\overline{U_i} = \mathtt{dc}(\mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, (\mathtt{T}(y,t) \ . \ u))$. By inspection of the rewrite rules, $\not\to^{\mathrm{H}} \overline{U_i}$, $\not\to^{\mathrm{H}} \mathtt{Nil}$, $\not\to^{\mathrm{H}} \mathtt{T}(y,t)$ and $\not\to^{\mathrm{H}} (\mathtt{T}(y,t) \ . \ u)$. By the properties of $\rightsquigarrow$ we then have $W = \mathtt{dc}(\mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, (x' \ . \ u'))$ where $x'{\downarrow} = \mathtt{T}(y,t)$ and $u'{\downarrow} = u$. Since $\mathrm{range}(\sigma_G)$ does not contain any $\mathtt{dc}$ symbols, we get that $U = C[\mathtt{dc}(\mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, \mathtt{Nil}, (x'' \ . \ u''))]$ for some C, x'', u'' such that $C_0 = C\sigma_G$ and $(x' \ . \ u') = (x'' \ . \ u'')\sigma_G$. Since U is in normal form, we must have $x'' \in \mathrm{dom}(\sigma_G)$ and thus $x' = \mathtt{T}(y,t) \in \mathrm{range}(\sigma_G)$, because otherwise $U \to$.
  By Lemma 3.2.6, we then have $\overline{U_i} \to^{\mathrm{H}} \in \mathcal{D}_{k+1}$, so $U_i \rightsquigarrow \in \mathcal{L}_{i+1}$.
  2 As 1 above.
  3 If $\overline{U_i} \to^h_{(3)}$, then $\overline{U_i} = \mathtt{dc}(v, \ w, \ x, \ (y \ . \ z), \ (\mathtt{T}(y,t) \ . \ u))$ for some x, y, z, t, u, v, w. We prove that $\overline{U_i}$ is in some $\mathcal{D}_k((\mathtt{T}(y,t) \ . \ u))$ by contradiction.
  - We may assume that this is the first time that rules (1-4) are applied to some redex not in $\mathcal{D}_l(\mathcal{T}_\Sigma)$. By induction, we have that redexes in $\mathcal{D}_l(\mathcal{T}_\Sigma) \cap \mathcal{L}_j$ only $\rightsquigarrow$-rewrite to terms in $\mathcal{D}_{l+1}(\mathcal{T}_\Sigma) \cap \mathcal{L}_{j+1}$ in two steps for $j < i$.
    By the properties of $\rightsquigarrow$, there are $x', y', y'', z', t', u', v', w' \in \mathcal{L}_0$ such that $U_0 \cong_C C_i[\mathtt{dc}(v', \ w', \ x', \ (y' \ . \ z'), \ (\mathtt{T}(y'',t') \ . \ u'))]$, $y' \rightsquigarrow^{i_1} y$ and $y'' \rightsquigarrow^{i_2} y$. By strong induction, we then have that $y' = y''$, so since σ_G is injective we also have $U \to$, which is a contradiction.

  We thus have $\overline{U_i} \in \mathcal{D}_k((\mathtt{T}(y,t) \ . \ u))$, and then $(y \ . \ z) = [\widetilde{n}]$ for some $\widetilde{n} \in X_G^*$, so specifically $y \in X_G$. By inspection of the rewrite rules, $\not\to^h (\mathtt{T}(y,t) \ . \ z)$, $\not\to^h \mathtt{T}(y,t)$, and $\not\to^h y$. By freshness constraints, we must have $\mathtt{T}(y,t) \in \mathrm{range}(\sigma_G)$. By Lemma 3.2.6, we then have $\overline{U_i} \to^h \in \mathcal{D}_{k+1}$, so $U_i \rightsquigarrow \in \mathcal{L}_{i+1}$.
  4 As 3 above.
  5 By inspection, $\mathcal{L}_i \not\to_{(5)}$.

  (II) Assume that $U'_0 \in \mathcal{L}_0$ such that $U'_0 \rightsquigarrow^* U'_i \in \mathcal{L}_i$ and $U'_i \rightsquigarrow^* U'_{i+1} \in \mathcal{L}_{i+1}$ as

above. Let C'_i be the greatest context of a redex $\overline{U_i}'$ such that $U'_i = C'_i[\overline{U_i}']$ and V, V' be such that $U_{i+1} = C_i[V]$ and $U'_{i+1} = C'_i[V']$. By (I), there exist $k, k' \geq 0$ and y, y' such that $\overline{U_i} \in \mathcal{D}_k(\mathcal{T}_\Sigma)$, $\overline{U_i}' \in \mathcal{D}_{k'}(\mathcal{T}_\Sigma)$, $V \in \mathcal{D}_{k+1}(y)$ and $V' \in \mathcal{D}_{k'+1}(y')$. We show that $C_i = C'_i$ by contradiction.

- Assume that $C'_i \neq C_i$. By symmetry we may assume that $C'_i < C_i$. By the properties of $\rightsquigarrow$, we then have (by induction) that $U_0 \cong_{C'_i} C'_i[V_0]$ for some $V_0 \in \mathcal{D}_{k'+1}(\mathcal{T}_\Sigma)$. Since σ_G does not contain any $\mathtt{dc}$ symbol we must have $V_0 \in \mathcal{L}_0$, but $\mathcal{L}_0 \cap \mathcal{D}_l = \emptyset$ for $l > 0$.

Since $C_i[V] = U_{i+1} = U'_{i+1} = C'_i[V']$, $C_i = C'_i$ gives that $V = V'$. Since the $\mathcal{L}_l$ are disjoint for different l, we also have $k = k'$. Since the $\mathcal{D}_k(x)$ are disjoint for different x, $y = y'$.

(a) If $k = 0$, we have by (I) that $\overline{U_i} \to^h_{(1,2)} V$ and $\overline{U_i}' \to^h_{(1,2)} V$. Since the rules (1,2) are injective, $\overline{U_i} = \overline{U_i}'$ and thus $U_i = U'_i$. By P(i-1), $U_0 = U'_0$.

(b) If $k > 0$, we assume that $V = \mathtt{dc}(v,\ dl(k+1),\ [\widetilde{a}],\ [\widetilde{n}],\ y)$. By induction (using the properties of $\rightsquigarrow$) there are $\widetilde{r}, \widetilde{r}'$ of length $k+1$ such that

$$U_{i-k} = C_i[\mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathrm{der}_y(\widetilde{r}))]$$
$$U'_{i-k} = C_i[\mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathrm{der}_y(\widetilde{r}'))]$$

By Lemma 3.2.6, $v \to^k_G \widetilde{a}\widetilde{n}$ using the partial leftmost derivations described by either of $\widetilde{r}$ and $\widetilde{r}'$. Since G is unambiguous and in CNF, we must have $\widetilde{r} = \widetilde{r}'$. Thus $\overline{U_i} = \overline{U_i}'$, so $U_i = U'_i$. By $P(i-1)$ we get $U_0 = U'_0$.

Given this, the statement of the lemma follows quickly.

1. Assume that $\sigma_G \vdash_{E_{\mathrm{CNF}}} U$ with U in normal form. Since equality is based on a convergent rewrite system and preserved by arbitrary substitution of terms for variables, we have that $\sigma_G \vdash_{E_{\mathrm{CNF}}} U$ iff there is $U' \in \mathcal{L}_0$ such that $U \equiv_{E_{\mathrm{CNF}}} U'$. By $\forall i \in \mathbb{N}.\ P(i)$, $U'{\downarrow} \in \mathcal{L}$, so $U \in \mathcal{L}$ by confluence.
2. Assume that $U_1, U_2 \in \mathcal{L}_0$ and $U_1 \equiv_{E_{\mathrm{CNF}}} U_2$. By definition there is V such that $V \not\rightarrow$, and $U_1 \rightsquigarrow^* V$ and $U_2 \rightsquigarrow^* V$. By $\forall i \in \mathbb{N}.\ P(i)$ there is k such that $V \in \mathcal{L}_k$, and $U_1 \rightsquigarrow^* V$ as by P. Since the $\mathcal{L}_k$ are disjoint for different k, we also have $U_2 \rightsquigarrow^* V$ as by P. $P(k-1)$ then yields $U_1 = U_2$.

□

Note that the statement of this lemma does not hold if G is ambiguous since in that case, two different elements in $\mathcal{L}_0$ can rewrite to the same term. For this reason, a similar characterization is hard to find in the general case. For instance, in the setting of [AC04b] it is often the case that two different terms (in the counterpart to our $\mathcal{L}_0$) can rewrite to the same term.

3.2.4 Reduction

We now know in sufficient detail how the grammar G relates to σ_G, and can proceed to the main result:

Theorem 3.2.10 *A grammar G in CNF is unambiguous iff $\sigma_G \cong_{E_{\mathrm{CNF}}} \rho_G$.*

Proof. As above, we write $G := (A_G, X_G, s_G, T_G \cup N_G)$.

- $\Leftarrow$ We prove the contrapositive of the implication from right to left. Assume that G is ambiguous. Then there exists $w \in A_G^*$ with two different leftmost derivations $\widetilde{r}^1$ and $\widetilde{r}^2$. Let $\mathrm{varOf}(k \to lm) := g_{\mathcal{V}}(k, l, m)$, $\mathrm{varOf}(n \to a) := f_{\mathcal{V}}(n, a)$ and $t_i := \mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ [\mathrm{varOf}(\widetilde{r}^i)])$ for $i = 1, 2$. By Lemma 3.2.6, we have that

$$t_1\sigma_G \to^* \mathtt{dc}(s_G,\ dl(1 + 2|w|),\ [w],\ \mathtt{Nil},\ \mathtt{Nil}) \text{ and}$$
$$t_2\sigma_G \to^* \mathtt{dc}(s_G,\ dl(1 + 2|w|),\ [w],\ \mathtt{Nil},\ \mathtt{Nil}),$$

  so $t_1\sigma_G = t_2\sigma_G$. By inspection, $t_1\rho_G \not\to$ and $t_2\rho_G \not\to$, so $t_1\rho_G \neq t_2\rho_G$. Thus σ_G and ρ_G are not statically equivalent.
- $\Rightarrow$ Assume that G is unambiguous. Let M and N be terms in normal form such that $\mathrm{n}(M, N) \cap \mathrm{n}(\mathrm{range}(\sigma_G) \cup \mathrm{range}(\rho_G)) = \emptyset$ and $(\mathrm{v}(M) \cup \mathrm{v}(N)) \subseteq \mathrm{dom}(\rho_G)$. Let $M_1 := M\sigma_G$, $M_2 := M\rho_G$, $N_1 := N\sigma_G$, and $N_2 := N\rho_G$.
  - Since ρ_G is injective, $\mathrm{range}(\rho_G)$ is in normal form, $\mathcal{N} \cap \mathrm{range}(\rho_G) = \emptyset$ and $\mathrm{range}(\rho_G)$ does not contain any function symbols that appear in rewrite rules, we have that M_2 and N_2 are in normal form. Then, by the injectivity of ρ_G, $M_2 \equiv_{E_{\mathrm{CNF}}} N_2$ implies that $M = N$, so $M_1 \equiv_{E_{\mathrm{CNF}}} N_1$.
  - Assume instead that $M_2 \not\equiv_{E_{\mathrm{CNF}}} N_2$. Then $M \neq N$, so by the injectivity of σ_G, we do not have $M_1 = N_1$. By Lemma 3.2.9, $M_1 \not\equiv_{E_{\mathrm{CNF}}} N_1$.

□

Corollary 3.2.11 *Since the ambiguity problem for context-free grammars is undecidable, $\cong_{E_{\mathrm{CNF}}}$ is undecidable.*

Example 3.2.12 *The grammar G_{p} is ambiguous, so $\sigma_{G_{\mathrm{p}}}$ and $\rho_{G_{\mathrm{p}}}$ should not be statically equivalent. Indeed, let $t = \mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ [x_1, x_1, x_4, x_1, x_4])$ and $u = \mathtt{dc}(\mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ \mathtt{Nil},\ [x_1, x_1, x_1, x_4, x_4])$.*
Then $t\sigma_{G_{\mathrm{p}}}{\downarrow}_{E_{\mathrm{CNF}}} = \mathtt{dc}(\mathtt{S},\ dl(5),\ [\mathtt{a}, \mathtt{a}, \mathtt{a}],\ \mathtt{Nil},\ \mathtt{Nil}) = u\sigma_{G_{\mathrm{p}}}{\downarrow}$
but $t\rho_{G_{\mathrm{p}}}{\downarrow}_{E_{\mathrm{CNF}}} = t\rho_{G_{\mathrm{p}}} \neq u\rho_{G_{\mathrm{p}}} = u\rho_{G_{\mathrm{p}}}{\downarrow}_{E_{\mathrm{CNF}}}$.

In conclusion, we have showed that there exists a message language where the construction problem is decidable but the indistinguishability problem is not. Since $\vdash_E$ can be reduced to $\approx_E$ in the presence of encryption [AC04a] or hashing [AC06], this means that there is a price to pay for the more sophisticated indistinguishability-based definition of secrecy: Static equivalence *is* harder than knowledge!

3.3 Constructor-Destructor Languages

In the previous section, we have seen that even convergent rewrite systems where deduction is decidable can yield undecidable static equivalence. Abadi and Cortier [AC06] have proved that both of these problems are decidable for the class of sub-term convergent rewrite systems. However, for this class of systems, the set of contexts that need to be considered to give all possible rewritings from a given knowledge set is dependent on the contents of that set (cf. σ-patterns, Definition 3.3.18). We propose the use of a subclass of these languages, *constructor-destructor languages*, that enable a uniform definition of the notions of synthesis and analysis. This class generalizes the message language of Section 1.4 by allowing arbitrary constructor symbols with matching destructor symbols.

Definition 3.3.1 *We work with a signature* $\Sigma = (\mathcal{F}^+ \uplus \mathcal{F}^-, \mathrm{ar})$ *where we split the set of function symbols* $\mathcal{F}$ *into* constructors $f \in \mathcal{F}^+$ *and* destructors $\overline{f} \in \mathcal{F}^-$. *Messages* $M, N \in \mathcal{M}$ *only contain constructor symbols (i.e., they are terms over* $(\mathcal{F}^+, \mathrm{ar}\,|_{\mathcal{F}^+})$*), while expressions* $F, G \in \mathcal{E} := \mathcal{T}_\Sigma$ *can also contain destructors. We assume that for all destructors* $\overline{f}$*, we have* $\mathrm{ar}(\overline{f}) \geq 1$.

We label the constructors with integers as f_i. *Each destructor is associated to a single constructor in the following way: For every destructor* $\overline{f}$ *there is exactly one rewrite rule, that is of the form* $\overline{f}(f_i(\widetilde{M}), \widetilde{N}) \to_{\mathrm{E}} M'$, *where* $\mathrm{n}(\overline{f}(f_i(\widetilde{M}), \widetilde{N})) = \emptyset$, $|\widetilde{M}| = \mathrm{ar}(f_i)$, $|\widetilde{N}| + 1 = \mathrm{ar}(\overline{f})$ *and* $M' \in \{\widetilde{M}\} \cup \{\widetilde{N}\}$. *We let* $j = \min(\{l \mid M' = M_l\} \cup \{l + \mathrm{ar}(f_i) \mid M' = N_l\})$, *and uniquely label such a destructor as* f_{ijk}^{-1}, *writing* LHS_{ijk} *(*RHS_{ijk}*) for the left-hand (right-hand) side of its rewrite rule.*

In keeping with the operational flavor of this language, we define term evaluation as $\mathbf{e}(F) := F{\downarrow_E}$ *whenever* $G{\downarrow_E} \in \mathcal{M}$ *for all subterms* G *of* F, *i.e., we require all destructors in* F *to succeed.*

In the remainder of this Section, we assume a fix constructor-destructor language $\Sigma, \to_E$ where the constructors, destructors and the terms occuring in rewrite rules are labelled as above.

Note that any constructor-destructor language is subterm convergent. As a comparison, the *data term languages* of [BPV05] constrain rewrite rules to be of the form $\overline{f}(\widetilde{M}) \to x$ (where $x \in \mathrm{v}(\overline{f}(\widetilde{M}))$), yielding a special case of (possibly non-convergent) subterm languages. We conjecture that the techniques of the above-mentioned paper could be straightforwardly adapted to constructor-destructor message languages, yielding a translation into the pi-calculus that is fully abstract with respect to may-testing.

The unicity of destructor rules in constructor-destructor languages was chosen to syntactically ensure a well-defined (deterministic) notion of evaluation, as well as strengthening the correspondence results between concrete and symbolic operational

semantics in Chapter 4. The constraint on the depth of RHS_{ijk} in LHS_{ijk} was chosen to yield the partial commutativity results of Lemma 3.3.11 and related results.

Example 3.3.2 *The nondeterministic choice rules* $\mathtt{either}((x\ .\ y)) \to x$ *and* $\mathtt{either}((x\ .\ y)) \to y$ *cannot both be present in a constructor-destructor language, but are permitted in a data term language. They also do not yield a convergent rewrite system.*

On the other hand, the limited inverse rule $\mathtt{f}(\mathtt{g}(\mathtt{h}(x))) \to \mathtt{h}(x)$ *can be part of a constructor destructor language (if* $\mathtt{g}$ *and* $\mathtt{h}$ *are constructors and* $\mathtt{f}$ *a destructor), but is not permitted in a data term language.*

The idempotent rule $\mathtt{f}(\mathtt{f}(x)) \to \mathtt{f}(x)$ *or the self-inverse rule* $\mathtt{f}(\mathtt{f}(x)) \to x$ *can be part of a subterm-convergent rewrite system, but are not permitted in a constructor-destructor language nor a data term language.*

The parameterized choice rules $\mathtt{pick}((x\ .\ y), \mathtt{1}) \to x$ *and* $\mathtt{pick}((x\ .\ y), \mathtt{2}) \to y$ *are permitted in a data term language and yield a convergent rewrite system, but are not permitted in a constructor-destructor language (but see the definition of* π_1 *and* π_2 *below).*

As a running example language, we extend the shared-key cryptographic system of Section 1.4 with hashing ($\mathsf{H}(\cdot)$) and public-key encryption ($\mathrm{E}^+_\cdot(\cdot)$, $\mathrm{E}^-_\cdot(\cdot)$) and decryption ($\mathrm{D}^+_\cdot(\cdot)$, $\mathrm{D}^-_\cdot(\cdot)$). We also add primitive constructs for pairing and pair splitting ($(\cdot\ .\ \cdot)$, π_1, π_2), generalizing the possibility of the polyadic π-calculus to send several channel names atomically. Furthermore, we no longer prohibit compound keys.

Example 3.3.3 *The example language has* $\Sigma_{\mathrm{DY}} = (\mathcal{F}^+ \cup \mathcal{F}^-, \mathrm{ar})$ *where*
$\mathcal{F}^+ = \{\mathrm{E}, \mathrm{E}^+, \mathrm{E}^-, \mathsf{H}, (\cdot\ .\ \cdot)\}$, $\mathcal{F}^- = \{\mathrm{D}, \mathrm{D}^+, \mathrm{D}^-, \pi_1, \pi_2\}$,
$1 = \mathrm{ar}(\mathsf{H}) = \mathrm{ar}(\pi_1) = \mathrm{ar}(\pi_2)$ *and*
$2 = \mathrm{ar}(\mathrm{E}) = \mathrm{ar}(\mathrm{E}^+) = \mathrm{ar}(\mathrm{E}^-) = \mathrm{ar}((\cdot\ .\ \cdot)) = \mathrm{ar}(\mathrm{D}) = \mathrm{ar}(\mathrm{D}^+) = \mathrm{ar}(\mathrm{D}^-)$.
The rewrite system $\to_{\mathrm{DY}}$ *is given by the rules*

$$\begin{aligned}
\mathrm{D}_y(\mathrm{E}_y(x)) &\to x \\
\mathrm{D}^+_y(\mathrm{E}^+_{\mathsf{pub}(y)}(x)) &\to x \\
\mathrm{D}^-_{\mathsf{pub}(y)}(\mathrm{E}^-_y(x)) &\to x \\
\pi_1(x\ .\ y) &\to x \\
\pi_2(x\ .\ y) &\to y
\end{aligned}$$

Note that this language also is a data term language.

3.3.1 Hedges, Revisited

Since we use a richer message language than in Chapter 2, we will also need to extend the operations on hedges that were defined there. We also extend the domain of hedges to cover expressions, in order to enable reasoning on not fully evaluated expressions.

The notion of analysis becomes slightly more complicated in the current setting, since we do not constrain the arguments of destructors ("keys") to be names. For a given destructor f_{ijk}^{-1}, the rule ANA-ijk attempts to apply f_{ijk}^{-1} to both sides of a pair in the analysis, constructing "keys" from the material that already has been analyzed.

Definition 3.3.4 (Hedges) *A* hedge *is a subset of $\mathcal{E} \times \mathcal{E}$. We denote by $\mathbf{H}$ the set of all hedges. The* synthesis *$\mathcal{S}(h)$ of a hedge h is the smallest hedge containing h and satisfying the rules*

$$(\text{SYN})\ \frac{(F_j, G_j) \in \mathcal{S}(h) \text{ for } j \in \{1, \ldots, \mathrm{ar}(f_i)\}}{(f_i(\widetilde{F}), f_i(\widetilde{G})) \in \mathcal{S}(h)}$$

Let $\mathcal{S}^+(h) := \{(f_i(\widetilde{F}), f(\widetilde{G})) \mid f_i \in \mathcal{F}^+ \wedge (F_j, G_j) \in \mathcal{S}(h) \text{ for } j \in \{1, \ldots, \mathrm{ar}(f)\}\}$.

The analysis *$\mathcal{A}(h)$ of a hedge h is defined by mutual induction with an auxiliary set $\mathcal{SA}(h)$ by the following rules*

$$(\text{ANA-KNOWN})\ \frac{(F, G) \in h}{(F, G) \in \mathcal{A}(h)}$$

$$(\text{ANA-}ijk)\ \frac{\begin{array}{c}(f_i(\widetilde{F}), f_i(\widetilde{G})) \in \mathcal{A}(h) \\ (F'_l, G'_l) \in \mathcal{SA}(h) \text{ for } l \in \{1, \ldots \mathrm{ar}(f_{ijk}^{-1}) - 1\}\end{array} \quad \begin{array}{l} f_{ijk}^{-1}(f_i(\widetilde{F}), \widetilde{F}') \to_E^{\mathrm{H}} F_j \\ f_{ijk}^{-1}(f_i(\widetilde{G}), \widetilde{G}') \to_E^{\mathrm{H}} G_j \end{array}}{(F_j, G_j) \in \mathcal{A}(h)}$$

$$(\text{ANA-S-KNOWN})\ \frac{(F, G) \in \mathcal{A}(h)}{(F, G) \in \mathcal{SA}(h)}$$

$$(\text{ANA-S-}i)\ \frac{(F_j, G_j) \in \mathcal{SA}(h) \text{ for } j \in \{1, \ldots, \mathrm{ar}(f_i)\}}{(f_i(\widetilde{F}), f_i(\widetilde{G})) \in \mathcal{SA}(h)}$$

The irreducibles *$\mathcal{I}(\cdot)$ of a hedge are defined as*

$$\mathcal{I}(h) := \mathcal{A}(h) \setminus \mathcal{S}^+(\mathcal{A}(h))$$

We write $h \vdash F \leftrightarrow G$ for $(F, G) \in \mathcal{S}(h)$. If h is a hedge, we let $h^{\mathrm{T}} := \{\, (G, F) \mid (F, G) \in h \,\}$ and $\pi_i(h) := \{\, F_i \mid (F_1, F_2) \in h \,\}$ when $i \in \{1, 2\}$.

A hedge h is irreducible iff $h = \mathcal{I}(h)$.

The only purpose of the set $\mathcal{SA}$ is to ensure that $\mathcal{A}(H)$ is well-founded. If we replaced $\mathcal{SA}(h)$ by $\mathcal{S}(\mathcal{A}(h))$ in ANA-ijk the definition would no longer be inductive, since we would need to argue about the presence or absence of certain expression pairs in $\mathcal{A}(h)$ before applying the rule.

Lemma 3.3.5 *For all hedges h, $\mathcal{SA}(h) = \mathcal{S}(\mathcal{A}(h))$.*

Proof. By induction on derivations of $(F, G) \in \mathcal{SA}(h)$ and $(F, G) \in \mathcal{S}(\mathcal{A}(h))$. □

These definitions and the following results are trivially transferable to message sets (or substitutions) by working with the corresponding hedges.

Definition 3.3.6 *If $\kappa \subset \mathcal{M}$, we let $X(\kappa) \stackrel{\text{def}}{=} \pi_1(X(\mathrm{Id}_\kappa))$ for $X \in \{\mathcal{S}, \mathcal{S}^+, \mathcal{A}, \mathcal{I}\}$.*

Example 3.3.7 *We work with the constructor-destructor language Σ_{DY}, $\equiv_{\mathrm{DY}}$, and let $h = \{(\mathsf{pub}(k), \mathsf{pub}(k)), (\mathrm{E}_k^-((m\ .\ n)), \mathrm{E}_k^-((m\ .\ m))), (\mathrm{E}_k^-(m), \mathrm{E}_l^-(m))\}$.*
Applying Definition 3.3.4 to h with this language, we get
$\mathcal{A}(h) = h \cup \{((m\ .\ m), (m\ .\ n)), (m, m), (m, n)\}$ and $\mathcal{I}(h) = h \cup \{(m, m), (m, n)\}$.

Since $\mathcal{A}$ and $\mathcal{S}$ are defined point-wise they possess the following algebraic properties, that are useful in proofs.

Proposition 3.3.8 *$\mathcal{A}$ and $\mathcal{S}$ are idempotent monotonous expansions, i.e.*

1. *$\mathcal{A}(\mathcal{A}(h)) = \mathcal{A}(h)$ and $\mathcal{S}(\mathcal{S}(h)) = \mathcal{S}(h)$.*
2. *$\mathcal{A}(g \cup h) \supseteq \mathcal{A}(g) \cup \mathcal{A}(h) \supseteq \mathcal{A}(h)$ and $\mathcal{S}(g \cup h) \supseteq \mathcal{S}(g) \cup \mathcal{S}(h) \supseteq \mathcal{S}(h)$.*
3. *$h \subseteq \mathcal{A}(h)$ and $h \subseteq \mathcal{S}(h)$.*

As before, if h can generate g then h can generate all messages generated by g, and conversely.

Lemma 3.3.9 *$g \leq h$ iff $g \subseteq \mathcal{S}(h)$*

Proof: Assume that $g \subseteq \mathcal{S}(h)$. Since $\mathcal{S}$ is idempotent and monotonous, we get $\mathcal{S}(g) \subseteq \mathcal{S}(\mathcal{S}(h)) = \mathcal{S}(h)$.
Assume that $g \leq h$. Since $g \subseteq \mathcal{S}(g)$ we immediately get $g \subseteq \mathcal{S}(h)$. □

We can then show that the generalized definition of irreducibles still permits to create all messages that can be created by the analysis.

Lemma 3.3.10 *$\mathcal{I}(h) \gtrless \mathcal{A}(h) \geq h$*

Proof: By $\mathcal{I}(h) \subseteq \mathcal{A}(h) \supseteq h$ and the monotonicity of $\mathcal{S}$ we get $\mathcal{I}(h) \leq \mathcal{A}(h) \geq h$. Since $\mathcal{A}(h) \setminus \mathcal{I}(h) \subseteq \mathcal{S}(\mathcal{I}(h))$, we get $\mathcal{A}(h) \subseteq \mathcal{S}(\mathcal{I}(h))$, so $\mathcal{A}(h) \leq \mathcal{I}(h)$ by Lemma 3.3.9. □

Since our destructors only extract the direct arguments of the constructor, iterated application of $\mathcal{S}$ and $\mathcal{A}$ does not bring any new knowledge.

Lemma 3.3.11

1. $\mathcal{S}(\mathcal{A}(h)) = \mathcal{S}(\mathcal{A}(\mathcal{S}(h)))$
2. $\mathcal{S}(\mathcal{A}(h)) = \mathcal{A}(\mathcal{S}(\mathcal{A}(h)))$

PROOF:

1. By monotonicity of $\mathcal{A}$ and $\mathcal{S}$ we get $\mathcal{S}(\mathcal{A}(h)) \subseteq \mathcal{S}(\mathcal{A}(\mathcal{S}(h)))$. To prove $\mathcal{S}(\mathcal{A}(h)) \supseteq \mathcal{S}(\mathcal{A}(\mathcal{S}(h)))$, by Lemma 3.3.9 it suffices to prove $\mathcal{S}(\mathcal{A}(h)) \supseteq \mathcal{A}(\mathcal{S}(h))$. We show that $\mathcal{S}(\mathcal{A}(h)) \supseteq \mathcal{S}(h) \cup \mathcal{A}(h) \supseteq \mathcal{A}(\mathcal{S}(h))$. The left inclusion follows from monotonicity and expansion properties. We prove the right inclusion in two steps:
   (a) We first show that whenever we analyze a message pair in $\mathcal{S}^+(h)$, the one-step analysis lies in $\mathcal{S}(h)$. Assume that we derive $(F, G) \in \mathcal{A}(\mathcal{S}(h))$ by applying ANA-ijk to $(f_i(\widetilde{F}), f_i(\widetilde{G})) \in \mathcal{S}^+(h)$ and some $(F'_l, G'_l) \in \mathcal{SA}(\mathcal{S}(h))$. Then there is $j \in \{1, \ldots, \mathrm{ar}(f)\}$ such that $F = F_j$ and $G = G_j$. Thus $(F, G) \in \mathcal{S}(h)$.
   (b) We now show that whenever we analyze something in $\mathcal{A}(h)$, the result always is in $\mathcal{A}(h)$. Assume that we derive $(F, G) \in \mathcal{A}(\mathcal{S}(h))$ by applying ANA-ijk to $(f_i(\widetilde{F}), f_i(\widetilde{G})) \in \mathcal{A}(h)$ and some $(F'_l, G'_l) \in \mathcal{SA}(\mathcal{S}(h))$.
   Assume that at least one $(F'_l, G'_l) \notin \mathcal{S}(\mathcal{A}(h)) = \mathcal{SA}(h)$, and that this is the first time in the derivation that this holds. We derive a contradiction: By monotonicity and idempotence of $\mathcal{S}$, and $\mathcal{A}$ being an expansion, we get

$$\mathcal{S}(\mathcal{A}(h)) \subseteq \mathcal{S}(\mathcal{A}(h) \cup \mathcal{S}(h)) \subseteq \mathcal{S}(\mathcal{S}(\mathcal{A}(h) \cup h)) = \mathcal{S}(\mathcal{A}(h))$$

   Thus $\mathcal{S}(\mathcal{S}(h) \cup \mathcal{A}(h)) = \mathcal{S}(\mathcal{A}(h))$, and since by the assumption we have only derived message pairs in $\mathcal{S}(h) \cup \mathcal{A}(h)$ so far we get that all $(F'_l, G'_l) \in \mathcal{S}(\mathcal{A}(h))$, so $(F, G) \in \mathcal{A}(h)$.

   Thus, the one-step analysis of a message pair in $\mathcal{S}^+(h) \cup \mathcal{A}(h) = \mathcal{S}(h) \cup \mathcal{A}(h)$ is always itself contained in $\mathcal{S}(h) \cup \mathcal{A}(h)$, so by induction $\mathcal{A}(\mathcal{S}(h)) \subseteq \mathcal{S}(h) \cup \mathcal{A}(h)$.
2. By monotonicity of $\mathcal{A}$ we get $\mathcal{S}(\mathcal{A}(h)) \subseteq \mathcal{A}(\mathcal{S}(\mathcal{A}(h)))$. For the other inclusion, we seek to prove that $\mathcal{S}(\mathcal{A}(h)) = \mathcal{S}(\mathcal{A}(\mathcal{S}(\mathcal{A}(h)))) \supseteq \mathcal{A}(\mathcal{S}(\mathcal{A}(h)))$, where the right inclusion holds by $\mathcal{S}$ being an expansion. By idempotence of $\mathcal{A}$ we get $\mathcal{S}(\mathcal{A}(h)) = \mathcal{S}(\mathcal{A}(\mathcal{A}(h)))$, and by 1 we have $\mathcal{S}(\mathcal{A}(\mathcal{A}(h))) = \mathcal{S}(\mathcal{A}(\mathcal{S}(\mathcal{A}(h))))$.

□

Using this, we can show that the following results, previously seen in Chapter 2, also hold for the full class of message languages. In particular, the irreducible hedges are the unique $\subseteq$-minimal representatives of their equivalence classes modulo $\gtrcurlyeq$.

Lemma 3.3.12

1. *If* $g \leq h$ *then* $\mathcal{I}(g) \leq \mathcal{I}(h)$
2. *If* $g \gtrcurlyeq \mathcal{I}(h)$ *then* $g \supseteq \mathcal{I}(h)$.
3. *If* $\mathcal{I}(g) \gtrcurlyeq \mathcal{I}(h)$ *then* $\mathcal{I}(g) = \mathcal{I}(h)$
4. $\mathcal{I}(\mathcal{I}(h)) = \mathcal{I}(h)$
5. $\mathcal{I}(h \cup g) = \mathcal{I}(\mathcal{I}(h) \cup g)$

Proof:

1. Assume that $g \leq h$. We get

$$\begin{aligned} g &\subseteq \mathcal{S}(h) && \text{by Lemma 3.3.9} \\ \mathcal{A}(g) &\subseteq \mathcal{A}(\mathcal{S}(h)) && \text{by monotonicity of } \mathcal{A} \\ \mathcal{A}(g) &\leq \mathcal{A}(\mathcal{S}(h)) && \text{by monotonicity of } \mathcal{S} \\ \mathcal{A}(g) &\leq \mathcal{A}(h) && \text{by Lemma 3.3.11.1} \\ \mathcal{I}(g) &\leq \mathcal{I}(h) && \text{by Lemma 3.3.10.} \end{aligned}$$

2. By contradiction: Let $(F, G) \in \mathcal{I}(h) \setminus g$. Since $(F, G) \in \mathcal{I}(h) \leq g$ we have $(F, G) \in \mathcal{S}(g)$, so by the assumption $(F, G) \in \mathcal{S}^+(g)$. By Lemma 3.3.9 we have $g \subseteq \mathcal{S}(\mathcal{I}(h))$, so by monotonicity and idempotence $(F, G) \in \mathcal{S}^+(\mathcal{I}(h))$. By monotonicity $\mathcal{S}^+(\mathcal{I}(h)) \subseteq \mathcal{S}^+(\mathcal{A}(h))$, so $(F, G) \in \mathcal{S}^+(\mathcal{A}(h))$. But $\mathcal{I}(h) \cap \mathcal{S}^+(\mathcal{A}(h)) = \emptyset$, so we cannot have $(F, G) \in \mathcal{I}(h)$.
3. By 2 we get $\mathcal{I}(g) \subseteq \mathcal{I}(h)$ and $\mathcal{I}(g) \supseteq \mathcal{I}(h)$.
4. By definition and the monotonicity and idempotence of $\mathcal{A}$, $\mathcal{I}(h) \subseteq \mathcal{A}(\mathcal{I}(h)) \subseteq \mathcal{A}(\mathcal{A}(h)) = \mathcal{A}(h)$ and $\mathcal{I}(h) \cap \mathcal{S}^+(\mathcal{A}(h)) = \emptyset$. Then

$$\begin{aligned} \mathcal{I}(h) &= \mathcal{I}(h) \setminus \mathcal{S}^+(\mathcal{A}(h)) \\ &\subseteq \mathcal{A}(\mathcal{I}(h)) \setminus \mathcal{S}^+(\mathcal{A}(h)) \\ &\subseteq \mathcal{A}(h) \setminus \mathcal{S}^+(\mathcal{A}(h)) \\ &= \mathcal{I}(h), \end{aligned}$$

   so $\mathcal{I}(h) = \mathcal{A}(\mathcal{I}(h)) \setminus \mathcal{S}^+(\mathcal{A}(h))$. We now seek to show that $\mathcal{A}(\mathcal{I}(h)) \setminus \mathcal{S}^+(\mathcal{A}(h)) = \mathcal{I}(\mathcal{I}(h))$. By Lemma 3.3.10 $\mathcal{A}(h) \gtrcurlyeq \mathcal{I}(h)$, so by 1 $\mathcal{I}(\mathcal{A}(h)) \gtrcurlyeq \mathcal{I}(\mathcal{I}(h))$. By Lemma 3.3.10 we get $\mathcal{A}(\mathcal{A}(h)) \gtrcurlyeq \mathcal{A}(\mathcal{I}(h))$, and by the idempotence of $\mathcal{A}$ we get $\mathcal{A}(h) \gtrcurlyeq \mathcal{A}(\mathcal{I}(h))$. Then $\mathcal{S}^+(\mathcal{A}(h)) = \mathcal{S}^+(\mathcal{A}(\mathcal{I}(h)))$, so $\mathcal{A}(\mathcal{I}(h)) \setminus \mathcal{S}^+(\mathcal{A}(h)) = \mathcal{A}(\mathcal{I}(h)) \setminus \mathcal{S}^+(\mathcal{A}(\mathcal{I}(h))) = \mathcal{I}(\mathcal{I}(h))$.
   Note that we do not have $\mathcal{A}(\mathcal{I}(h)) = \mathcal{I}(h)$ in all cases.
5. As Lemma 2.4.15.

□

3.3.2 Knowledge

Given these preliminary results, we can show that our definitions actually correspond to the notion of generatability of Section 3.1. The main difference is that concrete evaluation **e** requires that all subterms also evaluate to messages.

However, since the rewrite rules do not contain any destructor function symbols except at top level, we can safely replace all minimal subterms that do not normalize to messages by a fresh variable. The resulting expression may rewrite further than the original expression, and in particular always normalizes to a message.

Definition 3.3.13 *Take F, y and $\sigma : \mathcal{V} \rightharpoonup \mathcal{M}$ with $\mathrm{dom}(\sigma) \supseteq \mathrm{v}(F)$ and $y \notin \mathrm{dom}(\sigma)$. We let $R^y_\sigma(F)$ be defined inductively by the rules*

$$(\textsc{Rmsg})\ R^y_\sigma(M) = M$$

$$(\textsc{Rsyn})\ \frac{R^y_\sigma(F_j) = G_j \text{ for } j \in \{1, \ldots, \mathrm{ar}(f_i)\}}{R^y_\sigma(f_i(\widetilde{F})) = f_i(\widetilde{G})}$$

$$(\textsc{Rana-}ijk)\ \frac{R^y_\sigma(F_l) = G_l \text{ for } l \in \{1, \ldots, \mathrm{ar}(f^{-1}_{ijk})\}}{R^y_\sigma(f^{-1}_{ijk}(\widetilde{F})) = f^{-1}_{ijk}(\widetilde{G})}\ f^{-1}_{ijk}(\widetilde{G}\sigma)\!\downarrow \in \mathcal{M}$$

$$(\textsc{Rrepl-}ijk)\ \frac{R^y_\sigma(F_l) = G_l \text{ for } l \in \{1, \ldots, \mathrm{ar}(f^{-1}_{ijk})\}}{R^y_\sigma(f^{-1}_{ijk}(\widetilde{F})) = y}\ f^{-1}_{ijk}(\widetilde{G}\sigma)\!\downarrow \notin \mathcal{M}$$

Lemma 3.3.14 *If F, y and $\sigma : \mathcal{V} \rightharpoonup \mathcal{M}$ are such that $\mathrm{dom}(\sigma) \supseteq \mathrm{v}(F)$ and $y \notin \mathrm{dom}(\sigma) \cup \mathrm{v}(\mathrm{range}(\sigma))$, then $R^y_\sigma(F)\sigma\!\downarrow = R^y_\emptyset(F\sigma)\!\downarrow = R^y_\emptyset(F\sigma\!\downarrow)\!\downarrow \in \mathcal{M}$.*

Proof. By induction on the nesting level of destructors and the depth of constructor function symbols at top level of F. If $F \in \mathcal{M}$ then $F\sigma\!\downarrow = F\sigma \in \mathcal{M}$, so $R^y_\sigma(F)\sigma\!\downarrow = R^y_\emptyset(F\sigma)\!\downarrow = R^y_\emptyset(F\sigma\!\downarrow)\!\downarrow \in \mathcal{M}$.

If $F = f_i(\widetilde{F})$ with $R^y_\sigma(F_j) = G_j$ for $j \in \{1, \ldots, \mathrm{ar}(f_i)\}$, then by induction $R^y_\sigma(F_j)\sigma\!\downarrow = R^y_\emptyset(F_j\sigma)\!\downarrow = R^y_\emptyset(F_j\sigma\!\downarrow)\!\downarrow \in \mathcal{M}$ for all j. Then $f_i(\widetilde{G}\sigma\!\downarrow) = R^y_\sigma(F)\sigma\!\downarrow = R^y_\emptyset(F\sigma)\!\downarrow = R^y_\emptyset(F\sigma\!\downarrow)\!\downarrow \in \mathcal{M}$.

If $F = f^{-1}_{ijk}(\widetilde{F})$ with $R^y_\sigma(F_l) = G_l$ for $l \in \{1, \ldots, \mathrm{ar}(f^{-1}_{ijk})\}$, then by induction $R^y_\sigma(F_l)\sigma\!\downarrow = R^y_\emptyset(F_l\sigma)\!\downarrow = R^y_\emptyset(F_l\sigma\!\downarrow)\!\downarrow = N_l \in \mathcal{M}$ for all l. If $f^{-1}_{ijk}(\widetilde{G}\sigma)\!\downarrow \notin \mathcal{M}$ then

$R^y_\sigma(F)\sigma\!\downarrow = y = R^y_\emptyset(F\sigma)\!\downarrow$. Moreover,

$$\begin{aligned} f^{-1}_{ijk}(\widetilde{R^y_\emptyset(F_l\sigma\!\downarrow)\!\downarrow})\!\downarrow &= f^{-1}_{ijk}(\widetilde{R^y_\sigma(F_l)\sigma\!\downarrow})\!\downarrow \\ &= f^{-1}_{ijk}(\widetilde{G}\sigma\!\downarrow)\!\downarrow \\ &= f^{-1}_{ijk}(\widetilde{G}\sigma)\!\downarrow \notin \mathcal{M}, \end{aligned}$$

so we may apply RREPL-ijk to derive $R^y_\emptyset(F\sigma\!\downarrow) = y = y\!\downarrow$.

Otherwise, if $f^{-1}_{ijk}(\widetilde{G}\sigma)\!\downarrow = M_j \in \mathcal{M}$ with $G_1\sigma\!\downarrow = f_i(\widetilde{M})$ then $R^y_\sigma(F)\sigma = f^{-1}_{ijk}(\widetilde{G}\sigma) = R^y_\emptyset(F\sigma)$ by RANA-ijk, so $R^y_\sigma(F)\sigma\!\downarrow = M_j = R^y_\emptyset(F\sigma)\!\downarrow$. There are two cases for $F\sigma\!\downarrow = f^{-1}_{ijk}(\widetilde{F}\sigma\!\downarrow)\!\downarrow$. If $f^{-1}_{ijk}(\widetilde{F}\sigma\!\downarrow) \not\rightarrow^{\mathrm{H}}$ then

$$\begin{aligned} f^{-1}_{ijk}(\widetilde{R^y_\emptyset(F_l\sigma\!\downarrow)})\!\downarrow &= f^{-1}_{ijk}(\widetilde{R^y_\emptyset(F_l\sigma\!\downarrow)\!\downarrow})\!\downarrow \\ &= f^{-1}_{ijk}(\widetilde{R^y_\sigma(F_l)\sigma\!\downarrow})\!\downarrow \\ &= f^{-1}_{ijk}(\widetilde{G}\sigma\!\downarrow)\!\downarrow \\ &= f^{-1}_{ijk}(\widetilde{G}\sigma)\!\downarrow = M_j \in \mathcal{M} \end{aligned}$$

so we may apply RANA-ijk with $R^y_\emptyset(F\sigma\!\downarrow)\!\downarrow = M_j$.

If $f^{-1}_{ijk}(\widetilde{F}\sigma\!\downarrow) \rightarrow^{\mathrm{H}} F'_j$ with $F_1\sigma\!\downarrow = f_i(\widetilde{F'})$ we take $\rho : \mathrm{v}(\mathrm{LHS}_{ijk}) \rightarrow \mathcal{E}$ with $\mathrm{LHS}_{ijk}\,\rho = f^{-1}_{ijk}(\widetilde{F}\sigma\!\downarrow)$ where $\mathrm{v}(\mathrm{LHS}_{ijk}) \cap \mathrm{v}(\sigma, y) = \emptyset$. We then let $\rho' = \{x \mapsto R^y_\emptyset(\rho(x)) \mid x \in \mathrm{dom}(\rho)\}$. By induction $\mathrm{range}(\rho') \subset \mathcal{M}$. By repeated use of RSYN we have $f^{-1}_{ijk}(R^y_\emptyset(F_i\sigma\!\downarrow)\!\downarrow) = \mathrm{LHS}_{ijk}\,\rho'$ and $R^y_\emptyset(F'_j\sigma)\!\downarrow = \mathrm{RHS}_{ijk}\,\rho'$. By induction $R^y_\emptyset(F_i\sigma\!\downarrow)\!\downarrow = R^y_\sigma(F_i)\sigma\!\downarrow$, so $\mathrm{LHS}_{ijk}\,\rho' = f^{-1}_{ijk}(R^y_\sigma(F_i)\sigma\!\downarrow) \rightarrow^{\mathrm{H}} R^y_\sigma(F)\sigma\!\downarrow = \mathrm{RHS}_{ijk}\,\rho' = R^y_\emptyset(F\sigma\!\downarrow)\!\downarrow$. □

Corollary 3.3.15 *Let F, y, σ be as above.*
If $F\sigma\!\downarrow = M \in \mathcal{M}$, then $\mathbf{e}(R^y_\sigma(F)\sigma) = M$.

Example 3.3.16 *Let $F = \mathrm{D}_{\pi_1(a)}(\mathrm{D}^+_x(\mathrm{E}^+_z(\mathrm{E}_{\pi_1(a)}(m))))$ and $\sigma = \{^k/_x\}\{^{\mathsf{pub}(k)}/_z\}$. Then $R^y_\sigma(F) = \mathrm{D}_y(\mathrm{D}^+_x(\mathrm{E}^+_z(\mathrm{E}_y(m))))$ and $\mathbf{e}(R^y_\sigma(F)\sigma) = m = F\sigma\!\downarrow$, although $\mathbf{e}(F\sigma)$ is undefined.*

We can now show that the notions of irreducibles, analysis and synthesis accurately capture the notion of generatability of messages.

Theorem 3.3.17 *Let $\sigma : \mathcal{V} \rightharpoonup \mathcal{M}$ be injective and idempotent. For all $M \in \mathcal{M}$, $\sigma \vdash_E M$ iff $M \in \mathcal{S}(\mathcal{I}(\mathrm{range}(\sigma) \cup (\mathrm{n}(M) \setminus \mathrm{n}(\mathrm{range}(\sigma)))))$.*

PROOF: We let $\kappa = \mathrm{range}(\sigma) \cup (\mathrm{n}(M) \setminus \mathrm{n}(\mathrm{range}(\sigma)))$.

$\Rightarrow$ Assume that $\sigma \vdash M$. Then there is F such that $\mathrm{n}(F) \cap \mathrm{n}(\mathrm{range}(\sigma)) = \emptyset$, $\mathrm{v}(F) \subseteq \mathrm{dom}(\sigma)$ and $F\sigma{\downarrow}M$. We let $F' = R^y_\sigma(F)\{^x/_y\}$ for some $x \in \mathrm{dom}(\sigma)$, $y \notin \mathrm{v}(\sigma, \kappa)$. Then $\mathbf{e}(F'\sigma) = M\{^{\sigma(x)}/_y\} = M$.
We have $\mathcal{I}(\kappa) \gtrdot \mathcal{A}(\kappa) \geq \kappa$ by Lemma 3.3.10. We prove that $M \in \mathcal{S}(\mathcal{A}(\kappa))$ by induction on F'. If $F' \in \mathrm{dom}(\sigma) \cup (\mathrm{n}(M) \setminus \mathrm{n}(\mathrm{range}(\sigma)))$ then $F'\sigma \in \kappa$ and we are done.
If $F' = f_i(\widetilde{F})$ then $F'\sigma{\downarrow} = f_i(\widetilde{F}\sigma{\downarrow})$ and by induction $\{\widetilde{F}\sigma{\downarrow}\} \subset \mathcal{S}(\mathcal{A}(\kappa))$. By SYN we then get $f_i(\widetilde{F}\sigma{\downarrow}) \in \mathcal{S}(\mathcal{A}(\kappa))$.
If $F' = f^{-1}_{ijk}(\widetilde{F})$ then $f^{-1}_{ijk}(\widetilde{F}\sigma{\downarrow}) \to^{\mathrm{H}} F'\sigma{\downarrow}$ and by induction $\{\widetilde{F}\sigma{\downarrow}\} \subset \mathcal{S}(\mathcal{A}(\kappa))$. By ANA-$ijk$ $F'\sigma{\downarrow} \in \mathcal{A}(\mathcal{S}(\mathcal{A}(\kappa))) = \mathcal{S}(\mathcal{A}(\kappa))$, where the equality follows from Lemma 3.3.11.2.

$\Leftarrow$ Assume that $M \in \mathcal{S}(\mathcal{I}(\kappa)) = \mathcal{S}(\mathcal{A}(\kappa))$. We find F such that $\mathbf{e}(F\sigma) = M$ by induction on the derivation of $M \in \mathcal{S}(\mathcal{A}(\kappa))$. If $\sigma(x) = M$, we let $F = x$. If $M \in \mathcal{N} \setminus \mathrm{n}(\mathrm{range}(\sigma))$, we let $F = M$.
If $M = f_i(\widetilde{M}) \in \mathcal{S}(\mathcal{A}(\kappa))$ was derived using SYN, then by induction there are F_l with $\mathbf{e}(F_l\sigma) = M_l$ for all $l \in \{1, \ldots, \mathrm{ar}(f_i)\}$. Letting $F = f_i(F)$, we get $\mathbf{e}(F\sigma) = f_i(\mathbf{e}(\widetilde{F}\sigma)) = f_i(\widetilde{M}) = M$.
If $M \in \mathcal{A}(\kappa)$ was derived using ANA-ijk, then there are $f_i(\widetilde{M}) \in \mathcal{A}(\kappa)$ and $\widetilde{N}$ with $N_l \in \mathcal{S}(\mathcal{A}(\kappa))$ for all $l \in \{1, \ldots, \mathrm{ar}(f^{-1}_{ijk})\}$ and $f^{-1}_{ijk}(f_i(\widetilde{M}), \widetilde{N}) \to^{\mathrm{H}} M$. By induction there are G, F_l with $\mathbf{e}(G\sigma) = f_i(\widetilde{M})$ and $\mathbf{e}(F_l\sigma) = N_l$ for all $l \in \{1, \ldots, \mathrm{ar}(f^{-1}_{ijk})\}$. Letting $F = f^{-1}_{ijk}(G, \widetilde{F})$, we get $\mathbf{e}(F\sigma) = \mathbf{e}(f^{-1}_{ijk}(f_i(\widetilde{M}), \widetilde{N})) = M$.

□

3.3.3 Consistency

In order to define a notion of consistency for concrete hedges, we use the notion of a pattern for a rewrite rule, intuitively a more abstract version of the LHS of the rule. As an extension of patterns, σ-patterns also track the possibilities to generate subterms of known messages (in $\mathrm{range}(\sigma)$). The definition is carefully chosen in order for Lemma A.1.2.2 to hold.

Definition 3.3.18 *An expression $f^{-1}_{ijk}(\widetilde{M})$ is a pattern if $f^{-1}_{ijk}(\widetilde{M}) \not\to^{\mathrm{H}}$ and there are $\sigma : \mathcal{V} \to (\mathcal{M} \setminus \mathcal{V})$ with $f^{-1}_{ijk}(\widetilde{M})\sigma = \mathrm{LHS}_{ijk}$. If $f^{-1}_{ijk}(\widetilde{M})$ is a pattern and $\sigma, \rho : \mathcal{V} \rightharpoonup \mathcal{M}$, then $f^{-1}_{ijk}(\widetilde{M}\rho)$ is a σ-pattern whenever*

$$\mathrm{range}(\rho) \subseteq \{M \notin \mathcal{V} \mid \mathrm{n}(M) = \emptyset \wedge \mathrm{v}(M) \subseteq \mathrm{dom}(\sigma) \wedge \exists N \in \mathrm{range}(\sigma)\ M\sigma \prec N\}$$

Example 3.3.19 *Modulo renaming of variables, the patterns for our example rewrite system are $\pi_1(x)$, $\pi_2(x)$, $\mathrm{D}_y(x)$, $\mathrm{D}^+_y(x)$, $\mathrm{D}^-_y(x)$, $\mathrm{D}^+_x(\mathrm{E}^+_z(y))$, $\mathrm{D}^-_x(\mathrm{E}^-_z(y))$ and $\mathrm{D}^-_{\mathsf{pub}(x)}(y)$.*

Compared to the previous definition (2.1.16) of consistency, we split the third condition into one construction and one destruction constraint (3 and 4 below).

Definition 3.3.20 *An irreducible hedge $h \subset \mathcal{M} \times \mathcal{M}$ is left consistent iff*

1. *if $(a, N) \in h$ with $a \in \mathcal{N}$ then $N \in \mathcal{N}$; and*
2. *if $(M, N), (M', N') \in h$ such that $M = M'$ then $N = N'$; and*
3. *if $(M, N) \in h$ there is no N' with $(M, N') \in \mathcal{S}^+(h)$; and*
4. *Take σ_1, σ_2 with $h = \{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\}$ and $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2)$. If $f_{ijk}^{-1}(\widetilde{M})$ is a σ_1-pattern and $f_{ijk}^{-1}(\widetilde{M})\sigma_1 \to^{\mathrm{H}}$ then $f_{ijk}^{-1}(\widetilde{M})\sigma_2 \to^{\mathrm{H}}$.*

h is consistent iff h and h^{T} are both left consistent.

Since there are only finitely many σ-patterns for any given σ, consistency is decidable.

Example 3.3.21 *Continuing Example 3.3.7, we let $h = \{(\mathsf{pub}(k), \mathsf{pub}(k)), (\mathrm{E}_k^-((m\ .\ n)), \mathrm{E}_k^-((m\ .\ m))), (\mathrm{E}_k^-(m), \mathrm{E}_l^-(m))\}$ and $g = \mathcal{I}(h) = h \cup \{(m, m), (m, n)\}$.*

Then g violates condition 2 for consistency since $h \supset \{(m, m), (m, n)\}$. g also violates condition 4 for consistency since $\mathrm{E}_k^-(m)$, but not $\mathrm{E}_l^-(m)$, can be decrypted by $\mathsf{pub}(k)$. Moreover, $\mathcal{I}(g \cup \{(k, k)\})$ violates condition 3, since $(\mathrm{E}_k^-(m), \mathrm{E}_l^-(m)) \in \mathcal{S}^+(\mathcal{I}(g \cup \{(k, k)\}))$.

When extending a (left) consistent hedge with fresh names, the resulting hedge is also (left) consistent.

Lemma 3.3.22 *If h is left consistent and $B \subset \mathcal{N}$ with $\mathrm{n}(h) \cap B = \emptyset$ then $h \cup B$ is left consistent.*

Proof: First, g is irreducible, since $\mathcal{A}(g) \cap \mathcal{S}^+(\mathcal{A}(g)) = \mathcal{A}(h) \cap \mathcal{S}^+(\mathcal{A}(h))$ and $\mathcal{A}(g) = \mathcal{A}(h) \cup \mathrm{Id}_{\mathrm{n}(F)}$. Then, conditions 1,2 and 3 of definition 3.3.20 trivially follow. Condition 4 holds since the applicability of rewrite rules in E is preserved by arbitrary substitution of terms for names. □

We show the validity of this notion of consistency in two ways. Firstly, it corresponds to static equivalence in an augmented message algebra, where we can directly check for each rewrite rule if it can be applied.

Definition 3.3.23 *Let $\Sigma = (\mathcal{F}^+ \uplus \mathcal{F}^-, \mathrm{ar})$ and E be as above. We define a new signature Σ' and equivalence relation E' based on a rewrite system $\to_{\mathrm{E}'}$ as follows. We let $\Sigma' = (\mathcal{F}^+ \uplus \mathcal{F}^- \uplus \mathcal{F}^?, \mathrm{ar}')$ where $\mathcal{F}^? \stackrel{\mathrm{def}}{=} \{\mathtt{name}, \mathtt{OK}\} \uplus \{f_{ijk}^? \mid f_{ijk}^{-1} \in \mathcal{F}^-\}$ and $\mathrm{ar}' = \mathrm{ar} \cup \{\mathtt{name} \mapsto 1, \mathtt{OK} \mapsto 0\} \cup \{f_{ijk}^? \mapsto \mathrm{ar}(f_{ijk}^{-1}) + 1\}$. We also let $\to_{\mathrm{E}'}$ be the extension of $\to_{\mathrm{E}}$ with the rules $f_{ijk}^?(f_i(\widetilde{M}), \widetilde{N}, \mathtt{OK}) \to_{\mathrm{E}'} \mathtt{OK}$ whenever $f_{ijk}^{-1}(f_i(\widetilde{M}), \widetilde{N}) \to_{\mathrm{E}}$, and the rule $\mathtt{name}(x) \to_{\mathrm{E}'} x$ when $x \in \mathcal{N}$.*

The correspondence theorem then looks as follows.

Theorem 3.3.24 *If* $\sigma_1, \sigma_2 : \mathcal{V} \rightharpoonup \mathcal{M}$ *with* $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2)$*, then* $\sigma_1 \cong_{E'} \sigma_2$ *iff* $\mathcal{I}(\{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\})$ *is consistent.*

PROOF: When $\mathcal{I}(\{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\})$ is consistent we get that $\sigma_1 \cong_{E'} \sigma_2$ by extending the result of Theorem 3.3.17 to the present setting. We treat the case where $\mathcal{I}(\{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\})$ is not left consistent by deriving expressions that are equated by σ_1 but distinguished by σ_2. The proof can be found in Appendix A.1. □

Secondly, precisely the same formulae are validated by the different sides of a consistent hedge.

Theorem 3.3.25 *If* $\sigma_1, \sigma_2 : \mathcal{V} \rightharpoonup \mathcal{M}$ *with* $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2)$ *and* $h = \{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\}$*, then* $\mathcal{I}(h)$ *is consistent iff for all* ϕ *satisfying* $\mathrm{n}(\phi) \cap \mathrm{n}(\mathrm{range}(\sigma_1) \cup \mathrm{range}(\sigma_2)) = \emptyset$ *and* $\mathrm{v}(\phi) \subseteq \mathrm{dom}(\sigma_1)$ *it holds that* $[\![\phi\sigma_1]\!]$ *iff* $[\![\phi\sigma_2]\!]$.

PROOF:

⇒: Assume that $\mathcal{I}(h)$ is consistent. Then $\sigma_1 \cong_{E'} \sigma_2$ by Theorem 3.3.24. Let $g = \mathcal{I}(h) \cup \mathrm{Id}_{\mathrm{n}(\phi)}$, which is consistent by Lemma 3.3.22. We prove the statement by induction on the structure of ϕ and its syntactic depth.

$[\, F : \mathcal{N} \,]$: Proof by contradiction: Assume that $[\![\phi\sigma_1]\!]$ and not $[\![\phi\sigma_2]\!]$.
Since $F\sigma_1{\downarrow_E} = F\sigma_1{\downarrow_{E'}} = \mathtt{name}(F\sigma_1){\downarrow_{E'}}$ we have $F\sigma_2{\downarrow_{E'}} = \mathtt{name}(F\sigma_2){\downarrow_{E'}}$ by $\sigma_1 \cong_{E'} \sigma_2$, so $F\sigma_2{\downarrow_{E'}} = F\sigma_2{\downarrow_E} \in \mathcal{N}$.
Since $\mathbf{e}(F\sigma_1)$ is defined, $G\sigma_1{\downarrow} \in \mathcal{M}$ for all $G \in \{G \mid G \prec F\}$. Thus $[\![[\, G = G \,]\sigma_1]\!]$, so by induction $[\![[\, G = G \,]\sigma_2]\!]$ and $\mathbf{e}(G\sigma_2) \in \mathcal{M}$. Then $\mathbf{e}(F\sigma_2)$ is defined and by definition equal to $F\sigma_2{\downarrow} \in \mathcal{N}$.

$[\, F = G \,]$: Proof by contradiction: Assume that $[\![\phi\sigma_1]\!]$ and not $[\![\phi\sigma_2]\!]$.
Since $F\sigma_1{\downarrow} = G\sigma_1{\downarrow}$ and $\sigma_1 \cong_{E'} \sigma_2$, we have $F\sigma_2{\downarrow} = G\sigma_2{\downarrow}$. Since $\mathbf{e}(F\sigma_1)$ is defined, $F_i\sigma_1{\downarrow} \in \mathcal{M}$ for all $F_i \in \{F_i \mid F_i \prec F\}$. Thus $[\![[\, F_i = F_i \,]\sigma_1]\!]$, so by induction $[\![[\, F_i = F_i \,]\sigma_2]\!]$ and $\mathbf{e}(F_i\sigma_2) \in \mathcal{M}$. Then $[\![F\sigma_2]\!]$ is defined and by definition equal to $F\sigma_2{\downarrow}$. Similarly, $[\![G\sigma_2]\!] = G\sigma_2{\downarrow}$.

$\neg\psi$: By induction $[\![\psi\sigma_1]\!]$ iff $[\![\psi\sigma_2]\!]$, so $[\![\neg\psi\sigma_1]\!]$ iff $[\![\neg\psi\sigma_2]\!]$.

$\psi_1 \wedge \psi_2$: By induction $[\![\psi_1\sigma_1]\!]$ iff $[\![\psi_1\sigma_2]\!]$, and $[\![\psi_2\sigma_1]\!]$ iff $[\![\psi_2\sigma_2]\!]$. Thus $[\![\psi_1 \wedge \psi_2\sigma_1]\!]$ iff $[\![\psi_1 \wedge \psi_2\sigma_2]\!]$.

⇐: By symmetry we assume that $\mathcal{I}(h)$ is not left consistent. There are four possibilities for left inconsistency of $\mathcal{I}(h)$.

1. If $(a, N) \in \mathcal{I}(h)$ with $a \in \mathcal{N}$ and $N \notin \mathcal{N}$, then there is $x \in \mathrm{dom}(\sigma_1)$ with $\sigma_1(x) = a$ and $\sigma_2(x) = N$. We let $\phi = \mathtt{name}(x)$.

2. If $(M,N),(M,N') \in \mathcal{I}(h)$ such that $N \neq N'$, then there are x, y with $\sigma_1(x) = M$, $\sigma_2(x) = N$, $\sigma_1(y) = M'$ and $\sigma_2(y) = N'$. We let $\phi = [\, x = y \,]$.
3. If $(M,N) \in \mathcal{I}(h)$ and $(M,N') \in \mathcal{S}^+(\mathcal{I}(h))$, then by monotonicity $\mathcal{S}^+(\mathcal{I}(h)) \subseteq \mathcal{S}^+(\mathcal{A}(h))$, and since $\mathcal{I}(h) \cap \mathcal{S}^+(\mathcal{A}(h)) = \emptyset$ we cannot have $(M,N) \in \mathcal{S}^+(\mathcal{I}(h))$. Thus $N \neq N'$.
   There is $x \in \mathrm{dom}(\sigma_1)$ with $\sigma_1(x) = a$ and $\sigma_2(x) = N$. By Lemma A.1.3 there is F with $\mathrm{n}(F) = \emptyset$, $(\mathbf{e}(F\sigma_1), \mathbf{e}(F\sigma_2)) = (M, N')$.
   We let $\phi = [\, x = F \,]$.
4. Otherwise, there is a σ_1-pattern $f_{ijk}^{-1}(\widetilde{M})$ with $f_{ijk}^{-1}(\widetilde{M})\sigma_1 \rightarrow^{\mathrm{H}} M'$ and $f_{ijk}^{-1}(\widetilde{M})\sigma_2 \not\rightarrow^{\mathrm{H}}$. We let $\phi = [\, f_{ijk}^{-1}(\widetilde{M}) = f_{ijk}^{-1}(\widetilde{M}) \,]$, where $[\![[\, f_{ijk}^{-1}(\widetilde{M}) = f_{ijk}^{-1}(\widetilde{M}) \,]\sigma_2]\!]$ does not hold since $\mathbf{e}(f_{ijk}^{-1}(\widetilde{M})\sigma_2)$ is undefined.

□

Example 3.3.26 *Continuing Example 3.3.21, we let* $\sigma_1 = \{^{\mathsf{pub}(k)}/_{x_1}\}\{^{\mathrm{E}_k^-((m\,.\,n))}/_{x_2}\}\{^{\mathrm{E}_k^-(m)}/_{x_3}\}$, $\sigma_2 = \{^{\mathsf{pub}(k)}/_{x_1}\}\{^{\mathrm{E}_k^-((m\,.\,m))}/_{x_2}\}\{^{\mathrm{E}_l^-(m)}/_{x_3}\}$ *and* $h = \{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\}$. *We have seen that* $\mathcal{I}(h)$ *is not consistent. The substitutions* σ_1 *and* σ_2 *are distinguished by the guards* $\phi = [\, \pi_1(\mathrm{D}_{x_1}^+(x_2)) = \pi_2(\mathrm{D}_{x_1}^+(x_2)) \,]$ *and* $\psi = [\, \mathrm{D}_{x_1}^+(x_3)) = \mathrm{D}_{x_1}^+(x_3)) \,]$*: We have* $[\![\phi\sigma_1]\!]$ *and* $[\![\psi\sigma_1]\!]$*, but neither* $[\![\phi\sigma_2]\!]$ *nor* $[\![\psi\sigma_2]\!]$ *hold.*

3.4 A Family of Spi Calculi

The difference between the spi calculus and the applied pi calculus from a modelling perspective is that the latter admits an arbitrary equivalence relation on expressions (subject to some type constraints), while the former distinguishes between expressions and messages, and has a notion of *evaluating* an expression to a message.

If messages model bit strings (or values of some ADT), expressions model recipes for computation on messages, and evaluation is then simply the deterministic execution of the expression. The evaluation of an expression may be undefined, signifying an exceptional situation such as type mismatch, pattern mismatch or failed checksum test.

In this section, we give some constraints on what we consider to be valid notions of evaluation, give some (non-)examples and redefine the spi calculus in this setting.

Definition 3.4.1 *Let* Σ *be a signature over* $\mathcal{N}$ *and* $\mathcal{V}$ *and* $\equiv_E$ *an equivalence on* $\mathcal{T}_\Sigma^c$*. We say that* $\mathbf{e} : \mathcal{T}_\Sigma^c \rightharpoonup \mathcal{T}_\Sigma^c$ *is an evaluation function for* Σ *and* $\equiv_E$ *if:*

1. $\mathbf{e}$ is idempotent: $\mathrm{range}(\mathbf{e}) \subseteq \mathrm{dom}(\mathbf{e})$ *and for all* $F \in \mathrm{range}(\mathbf{e})$, $F = \mathbf{e}(F)$.
2. All names are messages: $\mathcal{N} \subseteq \mathrm{range}(\mathbf{e})$.

3. $\mathbf{e}$ respects term equivalence: *If* $F \equiv_E G$ *with* $F, G \in \mathrm{dom}(\mathbf{e})$, *then* $F \equiv_E \mathbf{e}(F) = \mathbf{e}(G)$.
4. $\mathbf{e}$ does not invent names: *If* $F \in \mathrm{dom}(\mathbf{e})$ *and* $a \in \mathrm{n}(\mathbf{e}(F))$ *then* $a \in \mathrm{n}(F)$.
5. $\mathbf{e}$ is insensitive to renaming: *If* $\sigma : \mathcal{N} \to \mathcal{N}$ *is injective and* $F \in \mathrm{dom}(\mathbf{e})$, *then* $F\sigma \in \mathrm{dom}(\mathbf{e})$ *and* $\mathbf{e}(F\sigma) \equiv_E \mathbf{e}(F)\sigma$.
6. $\mathbf{e}$ can be applied recursively: *If* $\sigma : \mathrm{v}(F) \to \mathrm{dom}(\mathbf{e})$ *and* $\rho = \{x \mapsto \mathbf{e}(\sigma(x)) \mid x \in \mathrm{v}(F)\}$ *then*
   (a) $F\sigma \in \mathrm{dom}(\mathbf{e})$ *iff* $F\rho \in \mathrm{dom}(\mathbf{e})$*; and*
   (b) $\mathbf{e}(F\sigma) = \mathbf{e}(F\rho)$ *if both are defined.*

We then write M, N *for elements of* $\mathcal{M} := \mathrm{range}(\mathbf{e})$.

Example 3.4.2 *If* $\to_E$ *is a confluent rewrite system over* Σ *such that* $\forall a \in \mathcal{N}\ a \not\to_E$, *then* $\mathbf{e}(F) := F{\downarrow}$ *is an evaluation function for* $\Sigma, \equiv_E$.

1. $F{\downarrow} = F{\downarrow}{\downarrow}$.
2. $a{\downarrow} = a$ *by assumption.*
3. *If* $F \equiv_E G$ *then* $F \equiv_E F{\downarrow} = G{\downarrow}$.
4,5. *Since* $\mathrm{v}(t_2) \cup \mathrm{v}(\phi) \subseteq \mathrm{v}(t_1)$ *and* $\mathrm{n}(t_1) = \mathrm{n}(t_2) = \emptyset$ *for any rewrite rule "*$t_1 \to_E t_2$ *if* ϕ*".*
6. *By the confluence and closure under contexts of* $\to_E$.

Example 3.4.3 *If* $\Sigma, \equiv_E$ *is a constructor-destructor algebra, then* $\mathbf{e}$ *as defined in Definition 3.3.1 is an evaluation function for* Σ *and* $\equiv_E$.

1. $\mathbf{e}(M) = M$ *for all messages* M.
2. $\mathbf{e}(a) = a$ *since* a *is a message.*
3,4. *Since* $\to_E$ *is confluent, as above.*
5. *Assume that* $\mathbf{e}(F)$ *is defined. Since* $\mathrm{n}(\mathrm{LHS}_{ijk}) = \mathrm{n}(\mathrm{RHS}_{ijk}) = \emptyset$ *we get* $G\sigma{\downarrow} = G{\downarrow}\sigma \in \mathcal{M}$ *for all subterms* G *of* F, *and thus* $\mathbf{e}(F\sigma) = \mathbf{e}(F)\sigma$.
6. *By induction on* F.

   $F = a$: *In this case,* $\mathbf{e}(F\sigma) = \mathbf{e}(a) = a = \mathbf{e}(F\rho)$.

   $F = x$: *In this case,* $\mathbf{e}(F\sigma) = \mathbf{e}(\sigma(x)) = \rho(x) = F\rho = \mathbf{e}(F\rho)$.

   $F = f_i(\widetilde{F})$: *By induction, we have* $\mathbf{e}(F_l\sigma) = G_l$ *iff* $\mathbf{e}(F_l\rho) = G_l$ *for all* $\widetilde{G}$ *and* $l \in \{1, \ldots, \mathrm{ar}(f_i)\}$. *Then, iff all evaluations* $\mathbf{e}(F_l\sigma)$ *are defined, we get* $\mathbf{e}(F\sigma) = f_i(\widetilde{G}) = \mathbf{e}(F\rho)$.

   $F = f_{ijk}^{-1}(\widetilde{F})$: *By induction, we have* $\mathbf{e}(F_l\sigma) = G_l$ *iff* $\mathbf{e}(F_l\rho) = G_l$ *for all* $\widetilde{G}$ *and* $l \in \{1, \ldots, \mathrm{ar}(f_i)\}$. *Then* $\mathbf{e}(F\sigma) = \mathbf{e}(f_{ijk}^{-1}(\widetilde{G}))$ *is defined iff* $\mathbf{e}(F\rho) = \mathbf{e}(f_{ijk}^{-1}(\widetilde{G}))$ *is defined.*

A non-example, showing that the so-called data term languages [BPV05] do not in general admit an evaluation function.

Example 3.4.4 *The equivalence induced by the rewrite rules* $\texttt{either}((x\ .\ y)) \to x$ *and* $\texttt{either}((x\ .\ y)) \to y$ *can be part of a data term language but does not admit a notion of evaluation. Otherwise, we would need a* $\equiv_E \texttt{either}((a\ .\ b)) \equiv_E b$ *so* $\mathbf{e}(a) = \mathbf{e}(b)$ *by 4. By 2* $\mathbf{e}(a) = a$ *and* $\mathbf{e}(b) = b$*, so* $a = b$ *for all names* a *and* b*, which is a contradiction.*

The notion of evaluation is, however, broad enough to cover associative and commutative operators. We conjecture that every equivalence of [CDL06] can be equipped with a suitable evaluation function.

Example 3.4.5 *Let* $\Sigma = (\{(\cdot\ .\ \cdot)\}, \{(\cdot\ .\ \cdot) \mapsto 2\})$ *and let* $\equiv_E$ *be induced by the rewrite rules* $(x\ .\ y) \to (y\ .\ x)$ *and* $((x\ .\ y)\ .\ z) \to (x\ .\ (y\ .\ z))$*. We assume a total order* $<$ *on* $\mathcal{N}$ *and let* $\mathbf{e}$ *yield the left-associative expression where smaller names occur earlier (shallower) than greater names.*

We check one case of condition 5 on $F = (a\ .\ b)$*, to illustrate the choice of* $\equiv_E$ *rather than* $=$*. Assume that* $a < b$*, so that* $\mathbf{e}(F) = F$*. Let* $\sigma = \{a \mapsto b, b \mapsto a\}$*. Then* $\mathbf{e}(F)\sigma = F\sigma = (b\ .\ a) \equiv_E F = \mathbf{e}(F\sigma)$*.*

The following result is a variation of condition 5, showing that you need to evaluate messages again after (α-)renaming.

Lemma 3.4.6 *If* $\mathbf{e}$ *is an evaluation function for* $\Sigma, \equiv_E$*,* $\sigma : \mathcal{N} \to \mathcal{N}$ *is injective and* $F \in \mathrm{dom}(\mathbf{e})$*, then* $F\sigma \in \mathrm{dom}(\mathbf{e})$ *and* $\mathbf{e}(F\sigma) = \mathbf{e}(\mathbf{e}(F)\sigma)$*.*

PROOF: $\mathbf{e}(F) \in \mathrm{dom}(\mathbf{e})$ by 1, so $\mathbf{e}(F)\sigma \in \mathrm{dom}(\mathbf{e})$ by 5. Since $F \in \mathrm{dom}(\mathbf{e})$ 5 also yields that $\mathbf{e}(F\sigma) \equiv_E \mathbf{e}(F)\sigma$. We then get $\mathbf{e}(\mathbf{e}(F\sigma)) = \mathbf{e}(\mathbf{e}(F)\sigma)$ by 3 where $\mathbf{e}(\mathbf{e}(F\sigma)) = \mathbf{e}(F\sigma)$ by 1. □

The spi calculus is then defined as before, using any expression language with a notion of evaluation.

Definition 3.4.7 *Given a signature* Σ*, we let spi calculus guards and processes over* Σ *be given by the grammars in Table 1.1 (p. 11), with* F *and* G *interpreted as elements of* $\mathcal{T}_\Sigma$*.*

Assume that $\mathbf{e}$ *is an evaluation function for* Σ *and* $\equiv_E$*. We define* α*-equivalence on transitions as follows: If* $P \xrightarrow{(\nu \tilde{b})\,\overline{a}\,M} P'$*,* $\widetilde{c}$ *is a tuple of pair-wise different names with* $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(a, M) \cup \mathrm{fn}(P')) = \emptyset$ *and* $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ *is bijective then* $(P \xrightarrow{(\nu \tilde{b})\,\overline{a}\,M} P') =_\alpha (P \xrightarrow{(\nu \widetilde{c})\,\overline{a}\,\mathbf{e}(M\sigma)} P'\sigma)$*. We can then give a (concrete) operational semantics to the corresponding spi calculus following Table 1.4 (p. 13) and Table 1.5 (p. 14).*

One reason that we prefer to use a structural operational semantics with an explicit notion of evaluation over the reduction-style semantics of the applied pi calculus [AF01] is the following.

Example 3.4.8 *In the applied pi calculus, the processes* $\overline{a}\langle F\rangle$ *and* $\overline{a}\langle G\rangle$ *are structurally equivalent whenever* $F \equiv_{\mathrm{DY}} G$*. Letting* $F = \pi_1(b\,.\,c)$ *and* $G = \pi_1(b\,.\,x)$*, where* $F \equiv_{\mathrm{DY}} b \equiv_{\mathrm{DY}} G$*, we see that structural equivalence preserves neither the free names nor the free variables of a process.*

We also extend the notions of labelled (Definition 1.4.4) and hedged (Definition 2.1.19) bisimulation to this setting, by interpreting evaluation $\mathbf{e}$, irreducibles $\mathcal{I}$ and synthesis $\mathcal{S}$ as defined in Section 3.3. In the next chapter, we turn to the problem of efficiently verifying hedged bisimilarity using symbolic techniques.

Chapter 4

Symbolic Semantics for the Spi Calculus

In this chapter, we give a general symbolic operational semantics, not using auxiliary environments or explicit substitutions, for any spi calculus. We show that it corresponds to the concrete semantics, up to restriction. We also give a refinement for the case of a constructor-destructor expression language, where the set of names that will be extruded under different instantiations is unchanged and easy to compute. We also define a set of process environments that yield a fully abstract correspondence to the early labelled semantics of processes.

Moving to symbolic bisimilarity, we define a notion of symbolic environments that each corresponds to a set of hedges via instantiation. We define decompositions, notably the notion of full decomposition of an environment into environments with unique solutions. We also give an argument that infinite decompositions may be needed to achieve completeness.

We then proceed to define a corresponding notion of symbolic bisimulation. As discussed in the introduction, the bisimulation is significantly more demanding than a straightforward adaptation of existing approaches in less complex calculi. We finally prove that this bisimulation is sound and complete with respect to its concrete counterpart.

At an earlier stage [BBN04] of this work, we considered only a single fixed expression language and did not have a notion of decomposition, making the resulting bisimilarity incomplete with respect to the concrete case.

4.1 Symbolic Operational Semantics

The idea behind the symbolic operational semantics is to defer instantiation on input and the evaluation of expressions and guards. Name freshness conditions are

still expressed as side conditions of the derivation rules, but all other conditions are simply collected in *transition constraints*. These constraints are the same as in the concrete semantics, namely the following.

- Guards.
- Channel equality for internal communication.
- Expressions standing for transmitted messages should evaluate correctly.
- Expressions used as channels should evaluate to names, not to compound messages.

Our symbolic semantics is similar to the symbolic operational semantics for the pi-calculus by Boreale and De Nicola [BD96]. The main difference is that we do not remove restricted names from the transition label as the transition is derived, due to the complex expression language. Because of this, a given transition constraint may be unsatisfiable, on its own or in conjunction with earlier transition constraints, somthing which we resolve at the level of environment-sensitive bisimulations or traces. The form of a symbolic transition is $P \xrightarrow[\phi]{\mu_s} P'$, where μ_s is a symbolic action and ϕ is a constraint.

Definition 4.1.1 *We define symbolic actions* $\mu_s \in \mathcal{A}_\mathrm{s}$ *as* $\mu_s ::= (\nu\tilde{c})\,F(x) \mid (\nu\tilde{c})\,\overline{F}\,G \mid (\nu\tilde{c})\,\tau$. *We define* α*-equivalent symbolic transitions in the following way:* $(P \xrightarrow[\phi]{\mu_s} P') =_\alpha (P \xrightarrow[\phi\sigma]{\mu_s\sigma} P'\sigma)$ *if* $\sigma : \mathcal{N} \to \mathcal{N}$ *is injective such that* $\mathrm{spt}(\sigma) \subseteq \mathrm{bn}(\mu_s)$, $(\mathrm{range}(\sigma) \setminus \mathrm{dom}(\sigma)) \cap (\mathrm{n}(\phi) \cup \mathrm{fn}(P')) = \emptyset$, $\phi' = \phi\sigma$, $P'' = P'\sigma$ *and* $\mu_s\sigma$ *is* μ_s *with every (both free and bound) name* n *replaced by* $\sigma(n)$.

We define the channel of an action (where applicable) as $\mathrm{ch}((\nu\tilde{c})\,\tau) = \emptyset$, $\mathrm{ch}((\nu\tilde{c})\,F(x)) := \{F\}$ *and* $\mathrm{ch}((\nu\tilde{c})\,\overline{F}\,G) := \{F\}$. *Symbolic transitions* $P \xrightarrow[\phi]{\mu_s} P'$ *are defined inductively by the rules of Table 4.1.*

Compared to the concrete semantics, in the rules (Sout) and (Sinp) the expressions are constrained rather than evaluated. In the rule Scom-l we need to keep the restriction on the action, since the transition guard contains the restricted names. When we encounter a guard, then rule (Sguard) simply adds it to the transition constraint.

The resulting processes after concrete resp. symbolic transitions differ in which names are restricted. We make their relationship precise in the remainder of the Section. We illustrate this with an example.

Example 4.1.2 *Let* $P := (\nu b)\,\overline{a}\langle\pi_1(a\,.\,b)\rangle.P'$ *for some* P'. *Concretely,* $P \xrightarrow{\overline{a}\,a} (\nu b)\,P'$. *Symbolically we have that* $P \xrightarrow[[a:\mathcal{N}]\wedge[\pi_1(a\,.\,b):\mathcal{M}]]{(\nu b)\,\overline{a}\,\pi_1(a\,.\,b)} P'$, *where the processes after the step only differ in the restriction of the name* b. *Also note that the*

$$(\textsc{Sout})\ \overline{G}\langle F\rangle.P \xrightarrow[[G:\mathcal{N}]\wedge[F:\mathcal{M}]]{\overline{G}\,F} P \qquad (\textsc{Sinp})\ G(x).P \xrightarrow[[G:\mathcal{N}]]{G(x)} P$$

$$(\textsc{Srep})\ !\,G(x).P \xrightarrow[[G:\mathcal{N}]]{G(x)} P \mid !\,G(x).P \qquad (\textsc{Sguard})\ \frac{P \xrightarrow[\phi]{\mu_s} P'}{\phi' P \xrightarrow[\phi\wedge\phi']{\mu_s} P'}$$

$$(\textsc{Scom-l})\ \frac{P \xrightarrow[\phi_1]{(\nu\tilde{b})\,G(x)} P' \quad Q \xrightarrow[\phi_2]{(\nu\tilde{c})\,\overline{G'}\,F} Q'}{P \mid Q \xrightarrow[\phi_1\wedge\phi_2\wedge[G=G']]{(\nu\tilde{b}\tilde{c})\ \tau} P'\{^F/_x\} \mid Q'} \begin{array}{ll} \text{if} & \{\tilde{c}\}\cap \mathrm{fn}(P)=\emptyset \\ \text{and} & \{\tilde{b}\}\cap \mathrm{fn}(Q)=\emptyset \\ \text{and} & \{\tilde{c}\}\cap\{\tilde{b}\}=\emptyset \end{array}$$

$$(\textsc{Scom-r})\ \frac{P \xrightarrow[\phi_1]{(\nu\tilde{b})\,\overline{G}\,F} P' \quad Q \xrightarrow[\phi_2]{(\nu\tilde{c})\,G'(x)} Q'}{P \mid Q \xrightarrow[\phi_1\wedge\phi_2\wedge[G=G']]{(\nu\tilde{b}\tilde{c})\ \tau} P' \mid Q'\{^F/_x\}} \begin{array}{ll} \text{if} & \{\tilde{c}\}\cap \mathrm{fn}(P)=\emptyset \\ \text{and} & \{\tilde{b}\}\cap \mathrm{fn}(Q)=\emptyset \\ \text{and} & \{\tilde{c}\}\cap\{\tilde{b}\}=\emptyset \end{array}$$

$$(\textsc{Ssum-l})\ \frac{P \xrightarrow[\phi]{\mu_s} P'}{P+Q \xrightarrow[\phi]{\mu_s} P'} \qquad (\textsc{Ssum-r})\ \frac{Q \xrightarrow[\phi]{\mu_s} Q'}{P+Q \xrightarrow[\phi]{\mu_s} Q'}$$

$$(\textsc{Spar-l})\ \frac{P \xrightarrow[\phi]{\mu_s} P'}{P \mid Q \xrightarrow[\phi]{\mu_s} P' \mid Q}\ \text{if}\ (\mathrm{bn}(\mu_s))\cap \mathrm{fn}(Q)=\emptyset$$

$$(\textsc{Spar-r})\ \frac{Q \xrightarrow[\phi]{\mu_s} Q'}{P \mid Q \xrightarrow[\phi]{\mu_s} P \mid Q'}\ \text{if}\ (\mathrm{bn}(\mu_s))\cap \mathrm{fn}(P)=\emptyset$$

$$(\textsc{Sopen})\ \frac{P \xrightarrow[\phi]{\mu_s} P'}{(\nu a)\,P \xrightarrow[\phi]{(\nu a)\,\mu_s} P'}\ \text{if}\ (\mathrm{fn}(\mu_s)\cup \mathrm{n}(\phi))\ni a\notin \mathrm{bn}(\mu_s)$$

$$(\textsc{Sres})\ \frac{P \xrightarrow[\phi]{\mu_s} P'}{(\nu a)\,P \xrightarrow[\phi]{\mu_s} (\nu a)\,P'}\ \text{if}\ a\notin \mathrm{n}(\mu_s)\cup \mathrm{n}(\phi)$$

$$(\textsc{Salp})\ \frac{P \xrightarrow[\phi]{\mu_s} P'}{P \xrightarrow[\phi']{\mu'_s} P''}\ \text{if}\ (P \xrightarrow[\phi]{\mu_s} P') =_\alpha (P \xrightarrow[\phi']{\mu'_s} P'')$$

Table 4.1: Symbolic Operational Semantics

scope of the binder for b in the symbolic transition extends to both the transition constraint and the resulting process.

Let $Q = (\nu c, d)\, \neg[\, a = c\,]\pi_1((a\ .\ d))(x).Q'$. *Then* $Q \xrightarrow{a(x)} (\nu c, d)\, Q'$ *and* $P \mid Q \xrightarrow{\tau} (\nu b)\, P' \mid (\nu c, d)\, (Q'\{^{a}/_{x}\})$. *Symbolically,* $Q \xrightarrow[[\,\pi_1(a\,.\,d)\,:\mathcal{N}\,]\wedge\neg[\,a=c\,]]{(\nu c,d)\,\pi_1((a\ .\ d))(x)} Q'$ *and* $P \mid Q \xrightarrow[\phi]{(\nu b,c,d)\,\tau} P' \mid Q'\{^{\pi_1(a\,.\,b)}/_{x}\}$ *where* $\phi = [\, a = \pi_1(a\,.\,d)\,] \wedge [\, a : \mathcal{N}\,] \wedge [\,\pi_1(a\,.\,b) : \mathcal{M}\,] \wedge [\,\pi_1(a\,.\,d) : \mathcal{N}\,] \wedge \neg[\, a = c\,]$. *Here* $[\![\phi]\!]$ *(defined in Table 1.4) is true.*

Since the processes P and Q above are closed (assuming that P' and Q' are closed), the guards of symbolic counterparts of concrete transitions simply evaluate to true.

4.1.1 A Single Step

We can use the symbolic semantics to verify if certain assignments to input variables, represented by a substitution σ, enable a concrete transition. We do this by comparing the effects of applying the substitution before and after a transition, both on the resulting processes and the transition constraints. The problem with this approach is that the symbolic semantics allow the communication of non-message terms, which after substitution need to be evaluated to coincide with the messages that are communicated in the concrete semantics.

Example 4.1.3 *Now consider* $Q := \overline{a}\langle \pi_1(x)\rangle \mid a(y).\overline{a}\langle y\rangle$. *We can derive* $Q \xrightarrow[\phi]{\tau} \mathbf{0} \mid \overline{a}\langle \pi_1(x)\rangle =: Q'$ *with* $\phi := [\, a : \mathcal{N}\,] \wedge [\,\pi_1(x) : \mathcal{M}\,] \wedge [\, a : \mathcal{N}\,] \wedge [\, a = a\,]$.
We do not have $[\![\phi]\!]$, *but the substitution* $\sigma := \{^{(a\,.\,a)}/_{x}\}$ *enables the transition. Concretely,* $Q\{^{(a\,.\,a)}/_{x}\} \xrightarrow{\tau} \mathbf{0} \mid \overline{a}\langle a\rangle$, *but* $\mathbf{0} \mid \overline{a}\langle a\rangle \neq \mathbf{0} \mid \overline{a}\langle \pi_1(a\,.\,a)\rangle = Q'\sigma$.

To cope with the problem outlined above and the fact that the symbolic semantics extrude more names (Example 4.1.2), we introduce the partial order $>_{\mathrm{a}}$ ("more abstract than"), which would be a subset of structural equivalence in an applied pi-style semantics [AF01].

Definition 4.1.4 *We let* $>_{\mathrm{a}}$ *be the least reflexive and transitive precongruence on expressions, guards and processes satisfying*

1. $F\sigma >_{\mathrm{a}} M\sigma$ *whenever* $\mathbf{e}(F) = M$ *and* $\sigma : \mathcal{N} \to \mathcal{N}$ *is injective;*
2. $(\nu a)\,(\nu b)\, P >_{\mathrm{a}} (\nu b)\,(\nu a)\, P$*; and*
3. $(\nu a)\,(P \mid Q) >_{\mathrm{a}} ((\nu a)\, P) \mid Q$ *when* $a \notin \mathrm{fn}(Q)$*; and*
4. $(\nu a)\,(P \mid Q) >_{\mathrm{a}} P \mid ((\nu a)\, Q)$ *when* $a \notin \mathrm{fn}(P)$.

Example 4.1.5 *Relating the effects of substituting before and after the transition in Example 4.1.3, we have* $Q'\sigma = (\mathbf{0} \mid \overline{a}\langle \pi_1((a\,.\,a))\rangle.\, \mathbf{0}) >_{\mathrm{a}} (\mathbf{0} \mid \overline{a}\langle a\rangle.\, \mathbf{0})$.

Lemma 4.1.6

1. *If* $F >_{\mathrm{a}} G$ *then* $\mathbf{e}(F) = \mathbf{e}(G)$*; and*
2. *if* $\phi >_{\mathrm{a}} \psi$ *then* $[\![\phi]\!]$ *iff* $[\![\psi]\!]$*; and*
3. *if* $P >_{\mathrm{a}} Q$ *then* $\mathrm{fn}(P) \supseteq \mathrm{fn}(Q)$*.*

Proof:

1. By Lemma 3.4.6 and Definition 3.4.1.6.
2. By 1, since the base cases of $[\![\cdot]\!]$ are defined entirely in terms of $\mathbf{e}$.
3. By $\mathrm{n}(F) \supseteq \mathrm{n}(\mathbf{e}(F))$, $\mathrm{fn}((\nu a)\,(\nu b)\,P) = \mathrm{fn}((\nu b)\,(\nu a)\,P)$, and $\mathrm{fn}((\nu a)\,(P \mid Q)) = \mathrm{fn}(((\nu a)\,P) \mid Q)$ and $\mathrm{fn}((\nu a)\,(Q \mid P)) = \mathrm{fn}(Q \mid ((\nu a)\,P))$ when $a \notin \mathrm{fn}(Q)$.

□

The relation $>_{\mathrm{a}}$ is a labelled bisimulation (Definition 1.4.4).

Lemma 4.1.7 $>_{\mathrm{a}}$ *is a standard labelled bisimulation. More precisely, if* $P >_{\mathrm{a}} Q$ *then*

1. *If* $P \xrightarrow{\mu} P'$ *then there is* Q' *such that* $Q \xrightarrow{\mu} Q'$ *and* $P' >_{\mathrm{a}} Q'$*; and*
2. *if* $Q \xrightarrow{\mu} Q'$ *such that* $\mathrm{bn}(\mu) \cap \mathrm{fn}(P) = \emptyset$ *then there is* P' *such that* $P \xrightarrow{\mu} P'$ *and* $P' >_{\mathrm{a}} Q'$*; and*
3. $P\sigma >_{\mathrm{a}} Q\sigma$ *for all substitutions* $\sigma : \mathcal{V} \rightharpoonup \mathcal{M}$*.*

Proof: The proof is by induction on the derivation of $P >_{\mathrm{a}} Q$ and the transition of P (or Q). The core idea is that the only operations performed on expressions are $\mathbf{e}$ and $\mathrm{fn}(\cdot)$, and that name clashes always can be avoided using α-renaming. Using Lemma 4.1.6.3, we do not need to assume $\mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$ in condition 1 of this lemma. For the full proof, see Appendix A.2. □

We also need to be sure that the α-renaming of guards does not change the substitutions that validate them.

Lemma 4.1.8 *For all formulae* ϕ*, if* $\rho : \mathcal{N} \to \mathcal{N}$ *is injective and* $\sigma : \mathcal{V} \rightharpoonup \mathcal{M}$ *satisfies* $\mathrm{n}(\mathrm{range}(\sigma)) \cap (\mathrm{spt}(\rho) \cup \mathrm{range}(\rho)) = \emptyset$ *then* $[\![\phi\sigma]\!]$ *iff* $[\![\phi\rho\sigma]\!]$*.*

Proof: We prove the statement by induction on ϕ.

First, given ρ, σ, we choose an injective ρ' with $\rho' \circ \rho \supseteq \{a \mapsto a \mid a \in \mathrm{n}(\phi)\}$ and $\mathrm{n}(\mathrm{range}(\sigma)) \cap (\mathrm{spt}(\rho') \cup \mathrm{range}(\rho')) = \emptyset$. Since $\phi\sigma = (\phi\rho)\rho'\sigma$ we only need to show one direction, $[\![\phi\sigma]\!]$. Moreover, $\phi\rho\sigma = \phi\sigma\rho$ by the assumption that $\mathrm{n}(\mathrm{range}(\sigma)) \cap (\mathrm{spt}(\rho) \cup \mathrm{range}(\rho)) = \emptyset$.

The nontrivial cases are:

$[\,F\!:\!\mathcal{N}\,]$ Assume that $\mathbf{e}(F\sigma) = a \in \mathcal{N}$, and that $a\rho = b$. By Definition 3.4.1.5, we have $\mathbf{e}(F\sigma\rho) = G \equiv_E b$. By Definition 3.4.1.(2,1) $\mathbf{e}(b) = b$, so by Definition 3.4.1.(1,3) $G = \mathbf{e}(G) = \mathbf{e}(b) = b \in \mathcal{N}$.

$[\,F\!:\!\mathcal{M}\,]$ Assume that $\mathbf{e}(F\sigma) = G$. By Definition 3.4.1.5, we have that $\mathbf{e}(F\sigma\rho)$ is defined.

$[\,F_1 = F_2\,]$ Assume that $\mathbf{e}(F_1\sigma) = G = \mathbf{e}(F_2\sigma)$. By Definition 3.4.1.5, we have that $\mathbf{e}(F_1\sigma\rho) = G_1 \equiv_E G\rho$ and $\mathbf{e}(F_2\sigma\rho) = G_2 \equiv_E G\rho$. Since $G_1 \equiv_E G_2$, Definition 3.4.1.(1,3) gives that $\mathbf{e}(F_1\sigma\rho) = \mathbf{e}(F_2\sigma\rho)$.

□

Note that the above is not true for injective $\rho : \mathcal{N} \to \mathcal{M}$:

Example 4.1.9 *$[\![\neg[\,\pi_1(b)\!:\!\mathcal{M}\,]]\!]$, but we do not have $[\![\neg[\,\pi_1(a\,.\,a)\!:\!\mathcal{M}\,]]\!]$.*

Using this, we can now show the relationship between concretely substituting inputs and our symbolic semantics.

Definition 4.1.10 *We let $\mathbf{e}(\cdot) : \mathcal{A}_{\mathrm{s}} \rightharpoonup \mathcal{A}$ be the partial function defined as $\mathbf{e}((\nu\tilde{b})\,\tau) := \tau$, $\mathbf{e}((\nu\tilde{b})\,F(x)) := \mathbf{e}(F)(x)$ when $\{\tilde{b}\} \cap \mathrm{n}(\mathbf{e}(F)) = \emptyset$, and $\mathbf{e}((\nu\tilde{b})\,\overline{F}\,G) := (\nu\tilde{c})\,\overline{\mathbf{e}(F)}\;\mathbf{e}(G)$ when $\tilde{c}$ is a tuple of pairwise different names satisfying $\{\tilde{c}\} = \{\tilde{b}\} \cap \mathrm{n}(\mathbf{e}(G))$ and $\{\tilde{c}\} \cap \mathrm{n}(\mathbf{e}(F)) = \emptyset$.*

Lemma 4.1.11

1. *If $P \xrightarrow[\phi]{\mu_s} P_1$ and σ is idempotent and satisfies $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$, $[\![\phi\sigma]\!]$ and $\mathbf{e}(\mu_s\sigma) = \mu$ then there are $\widetilde{c}, P_2$ with $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu_s\sigma))\} \uplus \{\widetilde{c}\}$ and $P\sigma \xrightarrow{\mu} P_2$ with $(\nu\tilde{c})\,P_1\sigma >_{\mathrm{a}} P_2$.*
2. *If σ is idempotent and $P\sigma \xrightarrow{\mu} P_1$ with $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu) = \emptyset$ then there are $\widetilde{c}, P_2, \phi, \mu_s$ such that $\mathrm{bn}(\mu_s) = \{\widetilde{c}\} \uplus \mathrm{bn}(\mu)$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$, $P \xrightarrow[\phi]{\mu_s} P_2$ and $(\nu\tilde{c})\,P_2\sigma >_{\mathrm{a}} P_1$.*

Proof: By induction on the derivation of the transition (see A.2). □

Example 4.1.12 *Why $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ in the above? Consider $P = (\nu n)\,[\,n = x\,]\overline{a}\langle n\rangle.\,\mathbf{0}$. There is a substitution $\left\{{}^{n}/_{x}\right\}$ that validates the transition constraint of $P \xrightarrow[[\,a:\mathcal{N}\,]\wedge[\,n:\mathcal{M}\,]\wedge[\,n=x\,]]{(\nu n)\,\overline{a}\,n} \mathbf{0}$ but does not enable a lazy transition when applied to P, due to α-renaming ($n \in \mathrm{bn}(\mu_s)$).*

Example 4.1.13 *Recall the parameterized choice rules* $\mathtt{pick}((x\ .\ y),1) \to x$ *and* $\mathtt{pick}((x\ .\ y),2) \to y$*, and let* $P = (\nu b,c)\,\overline{a}\langle\mathtt{pick}((b\ .\ c),x)\rangle.P'$*. Then* $P \xrightarrow[\phi]{(\nu b,c)\,\overline{a}\,\mathtt{pick}((b\ .\ c),x)} P'$ *for some* ϕ*.*

Depending on whether we instantiate x *with* $\mathtt{1}$ *or* $\mathtt{2}$*, the corresponding concrete transitions are* $P\{^{1}/_{x}\} \xrightarrow{(\nu b)\,\overline{a}\,b} (\nu c)\,P'\{^{1}/_{x}\}$ *and* $P\{^{2}/_{x}\} \xrightarrow{(\nu c)\,\overline{a}\,c} (\nu b)\,P'\{^{2}/_{x}\}$*, where different names are extruded in the two transitions. Moreover, we have* $P\{^{a}/_{x}\} \xrightarrow{(\nu b,c)\,\overline{a}\,\mathtt{pick}((b\ .\ c),a)} P'\{^{a}/_{x}\}$*.*

A Refinement for Constructor-Destructor Languages

As seen in Example 4.1.2, the symbolic semantics may extrude the scope of more names than the concrete semantics. However, when working with a constructor-destructor expression language, we can compute exactly which names would be extruded by the concrete semantics, using a notion of "abstract evaluation". This abstract evaluation, $\mathbf{e}_{\mathrm{a}} : \mathcal{E} \to \mathcal{E}$, intuitively reduces a term wherever possible, without checking that e.g. decryption and encryption keys correspond.

Definition 4.1.14 *If* Σ, E *is a constructor-destructor message algebra, we define a new rewrite system* $\to_{\mathrm{A}}$ *on* $\mathcal{T}_{\Sigma}$ *through the rules* $f_{ijk}^{-1}(f_i(\tilde{x}),\tilde{y}) \to_{\mathrm{A}} z$ *where* $z = x_j$ *if* $j \leq \mathrm{ar}(f_i)$*, otherwise* $z = y_{j-\mathrm{ar}(f_i)}$*. We then let* $\mathbf{e}_{\mathrm{a}}(F) \stackrel{\mathrm{def}}{=} F{\downarrow_A}$*.*

We let the extruded names of an expression $\mathrm{en}(F)$ *be defined inductively by* $\mathrm{en}(a) = \{a\}$*,* $\mathrm{en}(x) = \emptyset$*,* $\mathrm{en}(f_{ijk}^{-1}(\widetilde{G})) = \emptyset$ *and* $\mathrm{en}(f_i(\widetilde{G})) = \cup_j \mathrm{en}(G_j)$*.*

Example 4.1.15 *Let* $F := \pi_1(x)$ *and* $\sigma := \{^{(a\,.\,a)}/_{x}\}$*. We have* $\mathbf{e}_{\mathrm{a}}(F) = \pi_1(x)$*,* $\mathbf{e}_{\mathrm{a}}(F)\sigma = \pi_1(a\,.\,a)$ *and* $\mathbf{e}_{\mathrm{a}}(F\sigma) = a$*.*

We then have $\mathbf{e}(F) = \mathbf{e}_{\mathrm{a}}(F)$ for all $F \in \mathrm{dom}(\mathbf{e})$, or in other words, $\mathbf{e}_{\mathrm{a}}$ extends $\mathbf{e}$ to the entire set of expressions. Moreover, abstract evaluation commutes with substitution (modulo concrete evaluation).

Lemma 4.1.16

1. *If* $F \in \mathrm{dom}(\mathbf{e})$ *then* $\mathbf{e}_{\mathrm{a}}(F) = \mathbf{e}(F)$*.*
2. *If* $\sigma : \mathrm{v}(F) \to \mathcal{M}$ *satisfies* $\mathbf{e}(F\sigma) = M$ *then* $\mathbf{e}(\mathbf{e}_{\mathrm{a}}(F)\sigma) = M$ *and* $\mathrm{en}(\mathbf{e}_{\mathrm{a}}(F)) \subseteq \mathrm{n}(M) \subseteq \mathrm{en}(\mathbf{e}_{\mathrm{a}}(F)) \cup \mathrm{n}(\mathrm{range}(\sigma))$*.*

Proof:

1. For all expressions G, $G \to_A^{\mathrm{H}} G'$ whenever $G \to_E^{\mathrm{H}} G'$, so $F{\downarrow_E} = F{\downarrow_A}$ whenever $F{\downarrow_E} = M \in \mathcal{M}$ (since $M \not\to_{\mathrm{A}}$).

2. First, since $\rightarrow_A \cup \rightarrow_E$ is a convergent rewriting system preserved by substitution we have $\mathbf{e}(F\sigma) = \mathbf{e}(\mathbf{e}_{\mathrm{a}}(F)\sigma)$. We proceed by induction on $G = \mathbf{e}_{\mathrm{a}}(F)$.
   - If $G = a$ then $\mathrm{en}(G) = \{a\} = \mathrm{n}(\mathbf{e}(G\sigma))$.
   - If $G = x$ then $\mathrm{en}(G) = \emptyset$ and $\mathrm{n}(\mathbf{e}(G\sigma)) = \mathrm{n}(\sigma(x)) \subseteq \mathrm{n}(\mathrm{range}(\sigma))$.
   - If $G = f_i(\widetilde{G})$ then $\mathrm{en}(G) = \cup_j \mathrm{en}(G_j)$ and $\mathrm{n}(\mathbf{e}(G\sigma)) = \cup_j \mathrm{n}(\mathbf{e}(G_j\sigma))$. By induction $\mathrm{en}(G_i) \subseteq \mathrm{n}(\mathbf{e}(G_i)\sigma) \subseteq \mathrm{en}(G_i) \cup \mathrm{n}(\mathrm{range}(\sigma))$.
   - If $G = f^{-1}_{ijk}(\widetilde{G})$ then $\mathrm{en}(G) = \emptyset$. Since $G \not\rightarrow_A$, we have $j \leq \mathrm{ar}(f_i)$ and $G_1 \neq f_i(\widetilde{F})$ for all $\widetilde{F}$. Since $f^{-1}_{ijk}(\mathbf{e}(\widetilde{G}\sigma)) \rightarrow^{\mathrm{H}}_E \mathbf{e}(G_1\sigma) = f_i(\widetilde{M})$ for some $\widetilde{M}$. Then $G_1 \notin \mathcal{N}$ and $G_1 \neq f_l(\widetilde{F})$ for all $l, \widetilde{F}$, so $\mathrm{en}(G_1) = \emptyset$. By induction $\mathrm{n}(f_i(\widetilde{M})) \subseteq \mathrm{n}(\mathrm{range}(\sigma))$, so $\mathrm{n}(\mathbf{e}(G\sigma)) = \mathrm{n}(M_j) \subseteq \mathrm{n}(\mathrm{range}(\sigma))$.

□

Motivated by this lemma, we define a version of the symbolic transition system that adds back restrictions to the resulting process, yielding a stronger correspondence.

Definition 4.1.17 *We define the transition relation* $\xrightarrow[\phi]{\mu}_s$ *by*

$$\textsc{CDtau}\ \frac{P \xrightarrow[\phi]{(\nu\tilde{b})\,\tau} P'}{P \xrightarrow[\phi]{(\nu\tilde{b})\,\tau}_s (\nu\tilde{b})\,P'} \qquad \textsc{CDinp}\ \frac{P \xrightarrow[\phi]{(\nu\tilde{b})\,F(x)} P'}{P \xrightarrow[\phi]{(\nu\tilde{b})\,F(x)}_s (\nu\tilde{b})\,P'}\ \text{if } \{\tilde{b}\} \cap \mathrm{en}(\mathbf{e}_{\mathrm{a}}(F)) = \emptyset$$

$$\textsc{CDout}\ \frac{P \xrightarrow[\phi]{(\nu\tilde{c})\,\overline{F}\,G} P'}{P \xrightarrow[\phi]{(\nu\tilde{c})\,\overline{F}\,G}_s (\nu\tilde{b})\,P'}\ \begin{array}{lll} \text{if} & \widetilde{b} \text{ are pair-wise different} \\ \text{and} & \{\tilde{b}\} = \{\widetilde{c}\} \setminus \mathrm{en}(\mathbf{e}_{\mathrm{a}}(G)) \\ \text{and} & \{\tilde{c}\} \cap \mathrm{en}(\mathbf{e}_{\mathrm{a}}(F)) = \emptyset \end{array}$$

We then immediately get

Theorem 4.1.18

1. *If* $P \xrightarrow[\phi]{\mu_s}_s P_1$ *and* σ *is idempotent and satisfies* $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$, $[\![\phi\sigma]\!]$ *and* $\mu := \mathbf{e}(\mu_s\sigma)$ *is defined, then there is* P_2 *with* $P\sigma \xrightarrow{\mu} P_2$ *and* $P_1\sigma >_{\mathrm{a}} P_2$.
2. *If* σ *is idempotent and* $P\sigma \xrightarrow{\mu} P_1$ *with* $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu) = \emptyset$ *then there are* ϕ, μ_s, P_2 *such* $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$, $P \xrightarrow[\phi]{\mu_s}_s P_2$ *and* $P_2\sigma >_{\mathrm{a}} P_1$.

PROOF:

1. Assume that $P \xrightarrow[\phi]{\mu_s}_s P_1$ and σ is idempotent and satisfies $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ and $[\![\phi\sigma]\!]$. By definition there are $\widetilde{b}, P_1'$ such that $\{\tilde{b}\} \subseteq \mathrm{bn}(\mu_s)$, $P_1 = (\nu\tilde{b})\, P_1'$ and $P \xrightarrow[\phi]{\mu_s} P_1'$.
   Assume that $\mu = \mathbf{e}(\mu_s\sigma)$. By Lemma 4.1.16 we get that $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu_s\sigma))\} \uplus \{\widetilde{b}\}$. By Lemma 4.1.11 there are $\widetilde{c}, P_2$ with $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu_s\sigma))\} \uplus \{\widetilde{c}\}$ and $P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$ with $(\nu\tilde{c})\, P_1'\sigma >_{\mathrm{a}} P_2$. Since $\{\tilde{c}\} = \{\tilde{b}\}$ we get $P_1\sigma = (\nu\tilde{b})\, P_1'\sigma >_{\mathrm{a}} (\nu\tilde{c})\, P_1'\sigma$, so $P_1\sigma >_{\mathrm{a}} P_2$ by transitivity.
2. Assume that σ is idempotent and $P\sigma \xrightarrow{\mu} P_1$ with $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu) = \emptyset$.
   By Lemma 4.1.11 there are $\widetilde{c}, P_2, \phi, \mu_s$ such that $\mathrm{bn}(\mu_s) = \{\widetilde{c}\} \uplus \mathrm{bn}(\mu)$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$, $P \xrightarrow[\phi]{\mu_s} P_2$ and $(\nu\tilde{c})\, P_2\sigma >_{\mathrm{a}} P_1$.
   By definition there are $\widetilde{b}, P_1'$ such that $\{\tilde{b}\} \subseteq \mathrm{bn}(\mu_s)$, $P_1 = (\nu\tilde{b})\, P_1'$ and $P \xrightarrow[\phi]{\mu_s} P_1'$.
   By Lemma 4.1.16 we get that $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu_s\sigma))\} \uplus \{\widetilde{b}\}$.
   Thus $\{\tilde{c}\} = \{\tilde{b}\}$ and we get $P_1\sigma = (\nu\tilde{b})\, P_1'\sigma >_{\mathrm{a}} (\nu\tilde{c})\, P_1'\sigma$, so $P_1\sigma >_{\mathrm{a}} P_2$ by transitivity.

□

We now have a symbolic operational semantics that is sound and complete with respect to the concrete one (modulo $>_{\mathrm{a}}$, which is a labelled bisimulation) and is finitely branching (modulo choices of bound names and variables).

4.1.2 The Early Labelled Transition System

We now add the possibility of processes to receive messages from an outside observer that, like in the pi calculus, can use all the free names of the process. This simpler setting mainly serves to introduce some of the concepts of the symbolic bisimilarity.

Lemma 4.1.11 shows that the symbolic semantics work fine for modelling individual steps, but as in the pi calculus we need a slightly more complicated model to model sequences of transitions, in order to keep track of the ordering of scope extrusions and inputs. Each process will be associated with a process environment, intuitively containing a condensed account of the history of the process. In the process environment we will store previous transition constraints, input variables and extruded names. We must also store information related to when the scope of a given name was extruded, as highlighted in the following example.

Example 4.1.19 *Let $P := a(x).(\nu k)\,\overline{a}\langle k\rangle.[\,x = k\,]\overline{a}\langle a\rangle$. Since k is restricted in P, and comes only into existence after the input along a has been performed, it is*

impossible for the input variable x to have become equal (by substitution) to k. Now, P has a symbolic trace

$$P \xrightarrow[[a:\mathcal{N}]]{a\,x} s \xrightarrow[[a:\mathcal{N}]\wedge[k:\mathcal{M}]]{(\nu k)\,\overline{a}\,k} s \xrightarrow[[x=k]\wedge[a:\mathcal{N}]\wedge[a:\mathcal{M}]]{\overline{a}\,a} s\ \mathbf{0}$$

where the substitution $\{^k/_x\}$ satisfies the conjunction of the transition guards but does not correspond to a possible concrete trace, since k could not be input before *it was extruded.*

Since we (in this section) are only concerned with scope extrusions this problem could also be solved with *distinctions* following [MPW92]. As a preparation for the bisimulation environments to come, we will not pursue this direction, but instead use a process environment with explicit timing information. We use a process environment with three components.

1. A finite map $tn : \mathcal{N} \rightharpoonup \mathbb{N}$ containing bound names in messages sent by the process and the time they were sent.
2. A finite map $tv : \mathcal{N} \rightharpoonup \mathbb{N}$ containing input variables and the time they were input.
3. A constraint ψ assembled from all transition constraints up to this point.

Definition 4.1.20 *We define early input transitions as $P \xrightarrow{a\,M} Q$ iff $P \xrightarrow{a(x)} P'$ and there is $M \in \mathcal{M}$ such that $Q = P'\{^M/_x\}$. We define $\sigma@\mu_s$ where μ_s is a symbolic late input action such that $\mathrm{bn}(\mu_s) \cap \mathrm{n}(\mathrm{range}(\sigma)) = \emptyset$ as the early input action defined as follows: $\sigma@(\nu\tilde{b})\,\tau = \tau$, $\sigma@(\nu\tilde{b})\,\overline{F}\,G = (\nu\{b_i \mid b_i \in \mathrm{n}(\mathbf{e}(F\sigma))\})\,\overline{\mathbf{e}(F\sigma)}\ \mathbf{e}(G\sigma)$ and $\sigma@(\nu\tilde{b})\,F(x) = \mathbf{e}(F\sigma)\,x\sigma$.*

We write pe for the process environment (tn, tv, ψ), pe' for (tn', tv', ψ') and so on. A process environment pe is said to be well-formed *if $\mathrm{n}(\psi) \subseteq \mathrm{dom}(tn)$ and $\mathrm{v}(\psi) \subseteq \mathrm{dom}(tv)$.*

From here on, we only consider well-formed process environments. The role of the environment is to place constraints on substitutions σ instantiating input variables.

Definition 4.1.21 *$\sigma : \mathcal{V} \rightharpoonup \mathcal{M}$ respects the environment pe, written $pe \vdash \sigma$ if*

- $\mathrm{dom}(\sigma) \supseteq \mathrm{dom}(tv)$
- $[\![\psi\sigma]\!]$
- *If $v \in tv$ and $n \in \mathrm{n}(\sigma(v)) \cap \mathrm{dom}(tn)$, then $tn(n) < tv(v)$.*

We say that pe is possible *if $\exists\sigma : pe \vdash \sigma$.*

Using this notion of possibility, we can perform symbolic early input transitions as follows:

$$\text{(Sym-early)}\ \frac{\begin{array}{ll} P \xrightarrow[\phi]{\mu_s}_s P' & \\ tn' = tn \cup \{n \mapsto i+1 \mid n \in \mathrm{bn}(\mu_s)\} & \mathrm{bn}(\mu) \cap \mathrm{dom}(tn) = \emptyset \\ tv' = tv \cup \{x \mapsto i+1 \mid x \in \mathrm{bv}(\mu_s)\} & \mathrm{bv}(\mu) \cap \mathrm{dom}(tv) = \emptyset \\ \psi' = \psi \wedge \phi & pe' \text{ is possible} \\ i = \max(\mathrm{range}(tn) \cup \mathrm{range}(tv) \cup \{0\}) & \end{array}}{(pe, P) \xrightarrow{\mu_s} (pe', P')}$$

Note that constraints are accumulated, possibly further restricting the choice of previous inputs.

Lemma 4.1.22 *If* $(pe, P) \xrightarrow{\mu} (pe', P')$ *and* $pe' \vdash \sigma$ *then* $pe \vdash \sigma$.

Proof: By $\psi' = \psi \wedge \phi$, $tn' \supseteq tn$ and $tv' \supseteq tv$. □

Example 4.1.23 *If* $P := a(x).[\,x = a\,]a(y).\mathbf{0}$, *then the concrete semantics gives that* $P \xrightarrow{a(x)} [\,x = a\,]a(y).\mathbf{0}$, *which is not a closed process. Correspondingly, a symbolic transition of* P *is* $P \xrightarrow[[\,a:\mathcal{N}\,]]{a(x)}_s [\,x = a\,]a(y).\mathbf{0}$, *where* $[\![[\,a:\mathcal{N}\,]]\!]$ *holds. We then have* $[\,x = a\,]a(y).\mathbf{0} \xrightarrow[[\,a:\mathcal{N}\,]\wedge[\,x=a\,]]{a(y)}_s \mathbf{0}$, *where* $[\![[\,x = a\,]]\!]$ *is satisfied by* $\{{}^{a}/_{x}\}$.

Looking at the early input semantics, we have $P \xrightarrow{a\,a} [\,a = a\,]a(y).\mathbf{0}$, *where the resulting process is free to go on:* $[\,a = a\,]a(y).\mathbf{0} \xrightarrow{a\,a} \mathbf{0}$. *Starting from an empty process environment, we get*

$$((\emptyset, \emptyset, tt), P) \xrightarrow{a(x)} ((\emptyset, \{[x \mapsto 1]\}, [\,a:\mathcal{N}\,]), [\,x = a\,]a(y).\mathbf{0}).$$

Let $\phi := [\,a:\mathcal{N}\,] \wedge [\,x = a\,]$ *and* $tv := \{[x \mapsto 1]\}$.
Since $(\emptyset, tv, [\,a:\mathcal{N}\,] \wedge \phi) \vdash \{{}^{a}/_{x}\}$ *and* $[\![[\,x = a\,]\{{}^{a}/_{x}\}]\!]$ *we have*

$$((\emptyset, tv, [\,a:\mathcal{N}\,]), [\,x = a\,]a(y).\mathbf{0}) \xrightarrow{a(y)} ((\emptyset, tv \cup \{[y \mapsto 2]\}, [\,a:\mathcal{N}\,] \wedge \phi), \mathbf{0}).$$

The environment-sensitive semantics is sound and complete with respect to concrete early input semantics in the the sense that all instances of a symbolic trace correspond to a concrete trace, and vice versa.

Proposition 4.1.24

1. *If* $(pe, P) \xrightarrow{\tilde{\mu}} (pe', P')$ *and* $pe', \vdash \sigma$ *then* $P_i \xrightarrow{\sigma @ \mu_i} P_{i+1}$ *for* $0 \leq i \leq n$ *and* $P'\sigma >_{\mathrm{a}} P_{n+1}$, *where* $P_0 = P\sigma$ *and* $\tilde{\mu} = \mu_0\mu_1 \cdots \mu_n$.

2. *If* $\mathrm{fn}(P) \subseteq \mathrm{dom}(tn)$, $pe \vdash \sigma$ *and* $P\sigma \xrightarrow{\tilde{\mu}} P_{n+1}$ *such that* $(\mathrm{dom}(tn) \cup \mathrm{fn}(P\sigma)) \cap \mathrm{bn}(\tilde{\mu}) = \emptyset$ *and* $\mathrm{bn}(\mu_i) \cap \mathrm{bn}(\mu_j) = \emptyset$ *for* $0 \leq i < j \leq n$ *then* $(pe, P) \xrightarrow{\tilde{\mu}'} (pe', P')$ *such that* $\exists \rho : pe' \vdash \rho$, $\forall v \in \mathrm{dom}(tv) : \rho(v) = \sigma(v)$, $P'\rho >_{\mathrm{a}} P_{n+1}$ *and* $\mu_i = \sigma @ \mu'_i$ *for* $0 \leq i \leq n$ *where* $\tilde{\mu} = \mu_0\mu_1 \cdots \mu_n$ *and* $\tilde{\mu}' = \mu'_0\mu'_1 \cdots \mu'_n$.

Proof: By induction on n. For the full proof, see Appendix A.2. □

4.2 Symbolic Bisimulation

We now turn to the main topic of this chapter. In order to avoid the infinite branching of hedged bisimulation on process input, we define a notion of symbolic bisimulation that is sound and complete with respect to hedged (concrete) bisimulation, and thus with respect to barbed equivalence. The main ingredient of this definition is the symbolic environments that, as seen in Section 4.1.2, need to keep track of the accumulated transition constraints and the time relationships between inputs and outputs in order to make a proper accounting of the knowledge of the adversary.

4.2.1 Symbolic Environments

A symbolic environment is a concise description of a set of concrete environments, differing only in the instantiations of variables. Here, a *variable instantiation* is a pair of substitutions, that must *respect* the symbolic environment. The concrete environments that we get from instantiating variables in an environment-respecting way are called *concretizations*.

For symbolic bisimulation, the environments are similar to the ones used for symbolic early input. The main difference is that since we deal with two processes, we need to keep twice the amount of information. Of course, we still need to keep timing information relating knowledge and input variables. A symbolic environment consists of the following three elements.

1. A timed hedge $th : \mathcal{E} \times \mathcal{E} \rightharpoonup \mathbb{N}$ containing pairs of messages considered equal by the environment and the time they were received.
2. A timed variable set $tv : \mathcal{V} \rightharpoonup \mathbb{N}^+$ containing input variables and the time they were input.
3. A pair of restricted formulae $((\nu C)\,\phi, (\nu D)\,\psi)$ representing the accumulated transition constraints and sets of fresh names.

Definition 4.2.1 *We write se for the environment* $(th, tv, ((\nu C)\,\phi, (\nu D)\,\psi))$. *By abuse of notation, we write* ϕ *for* $(\nu\emptyset)\,\phi$ *and* $(\nu a)\,\phi$ *for* $(\nu\{a\})\,\phi$ *in environments. We let* $th^{\mathrm{T}} := \{(F,G)\mapsto th(G,F) \mid (G,F) \in \mathrm{dom}(th)\}$ *in order to swap the sides of a timed hedge. We let* $\mathrm{n}_1(se) := \mathrm{n}(\pi_1(\mathrm{dom}(th)))\cup C\cup\mathrm{n}(\phi)$, $\mathrm{n}_2(se) := \mathrm{n}(\pi_2(\mathrm{dom}(th)))\cup D\cup\mathrm{n}(\psi)$ *and* $\mathrm{n}(se) := \mathrm{n}_1(se)\cup\mathrm{n}_2(se)$.

Intuitively, if the environment knows the pair (F,G) it must have learned about it with the help of the processes at time $th(F,G)$; if this pair contains an input variable x, then the process must have performed this input at the strictly earlier time $tv(x)$.

Definition 4.2.2 *The environment se is* well-formed *if* $\mathrm{dom}(th)$ *is finite,* $0 \in \mathrm{range}(th)$, $\mathrm{v}(\mathrm{range}(th),\phi,\psi) \subseteq \mathrm{dom}(tv)$, *and whenever* $(F,G) \in \mathrm{dom}(th)$ *such that* $x \in \mathrm{v}(F,G)$ *then* $tv(x) < th(F,G)$.

From here on we only consider well-formed symbolic environments, the set of which is denoted **SE**.

By instantiating the input variables of the symbolic environment, we can get a concrete (non-timed) hedge. However, such an instantiation must be subject to several constraints, e.g., timing, guard satisfaction and freshness of invented names. For instance, an input performed at time t must be synthesizable from the knowledge of the environment at that time. Similarly to the symbolic early input semantics, we define environment respectfulness for substitutions. Naturally, with the bisimulation environments we need two (possibly different) substitutions, one for each process. We also create some fresh names B.

Definition 4.2.3 *A substitution pair* (σ,ρ) *is se-respecting with* $B \subseteq \mathcal{N}$, *written* $se \vdash \sigma \leftrightarrow_B \rho$ *iff*

1. $\mathrm{dom}(\sigma) = \mathrm{dom}(\rho) = \mathrm{dom}(tv)$; *and*
2. $[\![\phi\sigma]\!]$ *and* $[\![\psi\rho]\!]$; *and*
3. *if* $tv(x) = t$ *then* $(\sigma(x),\rho(x)) \in \mathcal{S}(\mathcal{I}(\{(F\sigma\downarrow, G\rho\downarrow) \mid th(F,G) \le t\} \cup \mathrm{Id}_B))$; *and*
4. B *is fresh and minimal in the sense that* $(\mathrm{n}(\mathrm{range}(th)) \cup C \cup D) \cap B = \emptyset$ *and if* $a \in B$ *then* $a \in \mathrm{n}(\mathrm{range}(\sigma))$ *or* $a \in \mathrm{n}(\mathrm{range}(\rho))$.

If $se \vdash \sigma \leftrightarrow_B \rho$ *we can concretize the knowledge th of the symbolic environment se by letting* $\mathbf{C}^B_{\sigma,\rho}(th) := \mathcal{I}(\{(\mathbf{e}(F\sigma), \mathbf{e}(G\rho)) \mid (F,G) \in \mathrm{dom}(th)\} \cup \mathrm{Id}_B)$.

In condition 3 of the above definition we use $F\sigma\downarrow$ rather than $\mathbf{e}(F\sigma)$ since the latter may be undefined. Indeed, $\mathbf{C}^B_{\sigma,\rho}(th)$ may be undefined, signifying that a received message was in fact a non-message expression. This cannot happen when using the symbolic semantics, since the requirement for the transmitted expression to be a message is always added to the transition constraint. This yields a *concretizable* symbolic environment (defined below), that always has well defined concretizations.

Example 4.2.4 *Let* $th := \{(x,x) \mapsto 2\}$, $tv := \{x \mapsto 1\}$ *and* $\sigma := \rho := \left\{{}^{\mathrm{E}_a(a)}/_{x}\right\}$. *Then we have* $(th, tv, (tt, tt)) \vdash \sigma \leftrightarrow_{\{a\}} \rho$, *and* $\mathbf{C}^{\{a\}}_{\sigma,\rho}(th) = \{(a,a)\}$ *is consistent. If the definition of* $\mathbf{C}^{\cdot}_{\cdot,\cdot}(\cdot)$ *did not use* $\mathcal{I}(\cdot)$, *then* $\mathbf{C}^{\{a\}}_{\sigma,\rho}(th) = \{(\mathrm{E}_a(a), \mathrm{E}_a(a)), (a,a)\}$ *would not be consistent.*

Since the *se*-respecting substitution pairs describe all admissible (with respect to the knowledge and constraints of *se*) instantiations of input variables, it is interesting to apply all of them to a pair of formulae (e.g., transition constraints) and study the results. If the formulae are only satisfied simultaneously, they are equivalent from the point of view of the environment. For an environment to be consistent, we require any concretization of its knowledge to be a consistent hedge. Moreover, we also require that the accumulated constraints are satisfied simultaneously on both sides (condition 2 below).

Definition 4.2.5 *We write* $se \vDash \phi' \Leftrightarrow \psi'$ *if for all* B, σ, ρ : $se \vdash \sigma \leftrightarrow_B \rho$ *implies that* $[\![\phi'\sigma]\!]$ *iff* $[\![\psi'\rho]\!]$. *se is* concretizable *if when* $(F,G) \in \mathrm{dom}(th)$ *we have* $se \vDash [F\!:\!\mathcal{M}] \Leftrightarrow tt$ *and* $se \vDash tt \Leftrightarrow [G\!:\!\mathcal{M}]$.

A concretizable symbolic environment se is consistent *if for all* B, σ, ρ,

1. $se \vdash \sigma \leftrightarrow_B \rho$ *implies that* $\mathbf{C}^B_{\sigma,\rho}(th)$ *is consistent; and*
2. $(th, tv, ((\nu C)\ tt, (\nu D)\ tt)) \vdash \sigma \leftrightarrow_B \rho$ *implies that* $[\![\phi\sigma]\!]$ *iff* $[\![\psi\rho]\!]$.

Note that if *se* is concretizable and $se \vdash \sigma \leftrightarrow_B \rho$ then $\mathbf{C}^B_{\sigma,\rho}(th)$ is always defined and σ and ρ are both idempotent.

When simulating a transition, we often need to consider different cases. In order to split a symbolic environment according to these cases, we may decompose the constraints [Bor95, HL95]. Since we keep constraints for both sides of the environment we may require that the split is consistent, following [JV07].

Definition 4.2.6 *Let* $se = (th, tv, ((\nu C)\,\phi, (\nu D)\,\psi))$ *be a concretizable symbolic environment. The set* $\{se_i\}_{i\in I}$ *is a decomposition of se if each* se_i *is consistent and of the form* $(th, tv, ((\nu C)\,\phi_i, (\nu D)\,\psi_i))$, *and whenever* $se \vdash \sigma \leftrightarrow_B \rho$ *there is* $i \in I$ *such that* $se_i \vdash \sigma \leftrightarrow_B \rho$. *A decomposition* $\{se_i\}_{i\in I}$ *is concretizable/consistent if each* se_i *is concretizable/consistent.*

Example 4.2.7 $\{se\}$ *is a decomposition of se.*

Let $se(\phi) := (\{(a,a) \mapsto 0\}, \{x \mapsto 1\}, (\phi, \phi))$. *Then* $\{se([\,x = a\,]), se(\neg[\,x = a\,])\}$ *is a decomposition of* $se(tt)$.

We can restrict a decomposition to the environments that have solutions, and we can combine decompositions in the following way.

Lemma 4.2.8

1. *If* $\{se_i\}_{i\in I}$ *is a decomposition of* se*, then* $S = \{se_i \mid i \in I \wedge \exists(\sigma, \rho, B)\ se_i \vdash \sigma \leftrightarrow_B \rho \wedge se \vdash \sigma \leftrightarrow_B \rho\}$ *is a decomposition of* se*. Moreover, if* $\{se_i\}_{i\in I}$ *is concretizable/consistent, then* S *also is.*
2. *If* $\{se_i\}_{i\in I}$ *and* $\{se_j\}_{j\in J}$ *are consistent decompositions of* se*, then* $S = \{se_{ij}\}_{(i,j)\in I\times J}$ *where* $\phi_{ij} = \phi_i \wedge \phi_j$ *and* $\psi_{ij} = \psi_i \wedge \psi_j$ *is a consistent decomposition of* se*.*

Proof:

1. Whenever $se \vdash \sigma \leftrightarrow_B \rho$ there is $i \in I$ such that $se_i \vdash \sigma \leftrightarrow_B \rho$. By definition $se_i \in S$.
2. Whenever $se_{ij} \vdash \sigma \leftrightarrow_B \rho$ there is $i \in I$ such that $se_i \vdash \sigma \leftrightarrow_B \rho$. $\mathbf{C}^B_{\sigma,\rho}(th)$ is then consistent by the consistency of se_i.
   Whenever $(th, tv, ((\nu C)\ tt, (\nu D)\ tt)) \vdash \sigma \leftrightarrow_B \rho$ we have that $[\![\phi_i\sigma]\!]$ iff $[\![\psi_i\rho]\!]$ by the consistency of se_i and and $[\![\phi_j\sigma]\!]$ iff $[\![\psi_j\rho]\!]$ by the consistency of se_j. Thus $[\![(\phi_i \wedge \phi_j)\sigma]\!]$ iff $[\![(\psi_i \wedge \psi_j)\rho]\!]$.
   Whenever $se \vdash \sigma \leftrightarrow_B \rho$ there is $i \in I$ such that $se_i \vdash \sigma \leftrightarrow_B \rho$ and $j \in J$ such that $se_j \vdash \sigma \leftrightarrow_B \rho$. Then $se_{ij} \vdash \sigma \leftrightarrow_B \rho$.

□

Every consistent environment can be decomposed into environments with unique solutions.

Definition 4.2.9 *A formula* ϕ *gives a unique solution for a finite set of variables* X *if there is exactly one substitution* $\sigma : X \to \mathcal{M}$ *such that* $[\![\phi\sigma]\!]$*. A consistent symbolic environment* $se = (th, tv, ((\nu C)\,\phi, (\nu D)\,\psi))$ *has the unique solution* (σ, ρ, B) *if* $se \vdash \sigma \leftrightarrow_B \rho$ *and* ϕ *and* ψ *both give a unique solution for* $\mathrm{dom}(tv)$*.*

We will mainly use the trivial/degenerated cases of this definition.

Example 4.2.10 *If* $\phi = \bigwedge_{x\in X}[\, x = M_x\,]$ *with* $\mathrm{v}(M_x) = \emptyset$ *for all* $x \in X$*, then* ϕ *gives a unique solution for* X*. If* h *is a consistent hedge with* $\mathrm{v}(h) = \emptyset$*, then* $se = (\{(F, G) \mapsto 0 \mid (F, G) \in h\}, \emptyset, ((\nu\emptyset)\ tt, (\nu\emptyset)\ tt))$ *has the unique solution* $(\emptyset, \emptyset, \emptyset)$*.*

We can *fully decompose* a consistent environment into an infinite set of environments with unique solutions as follows.

Lemma 4.2.11 *Let* $se = (th, tv, ((\nu C)\,\phi, (\nu D)\,\psi))$ *be a consistent environment and* $I = \{(\sigma, \rho, B) \mid se \vdash \sigma \leftrightarrow_B \rho\}$. *Then* $\{se_{(\sigma,\rho,B)}\}_{(\sigma,\rho,B)\in I}$ *where* $\phi_{(\sigma,\rho,B)} = \bigwedge_{x\in \mathrm{dom}(tv)}[\, x = \sigma(x)\,]$ *and* $\psi_{(\sigma,\rho,B)} = \bigwedge_{x\in \mathrm{dom}(tv)}[\, x = \rho(x)\,]$ *is a decomposition of* se, *and each* $se_{(\sigma,\rho,B)}$ *has the unique solution* (σ, ρ, B).

Proof: We need to show that each $se_{(\sigma,\rho,B)}$ is consistent and that $se_{(\sigma,\rho,B)} \vdash \sigma \leftrightarrow_B \rho$. Clearly, if $se_{(\sigma,\rho,B)} \vdash \sigma' \leftrightarrow_B \rho'$ then $\sigma = \sigma'$ and $\rho = \rho'$.

By assumption $(th, tv, ((\nu C)\ tt, (\nu D)\ tt)) \vdash \sigma \leftrightarrow_B \rho$. $\mathbf{C}^B_{\sigma,\rho}(th)$ is consistent by assumption, so by Lemma A.1.2. 1 $M = \sigma(x)$ (or $N = \rho(x)$) whenever $\mathbf{C}^B_{\sigma,\rho}(th) \vdash M \leftrightarrow \rho(x)$ (or $\mathbf{C}^B_{\sigma,\rho}(th) \vdash \sigma(x) \leftrightarrow N$, by symmetry). □

In the pi calculus, it is always sufficient to consider a finite number of cases in the decomposition [Bor95]. However, in a spi calculus an infinite split may be needed when treating processes with replication.

Example 4.2.12 *We take a simple expression language that allows us to encode integers. Let* $\Sigma = (\{\mathtt{s}\} \cup \{\mathtt{p}\}, \{\mathtt{s} \mapsto 1, \mathtt{p} \mapsto 1\})$ *and let* $\rightarrow_E$ *be induced by the single rewrite rule* $\mathtt{p}(\mathtt{s}(x)) \rightarrow x$. *This language is a constructor-destructor language, and would also be admissible as a data term language. We write* n_a *for the message* $\mathtt{s}^n(a)$.

We define two processes P *and* Q *with the same behavior (i.e.,* $P \sim Q$, *where* $\sim$ *is labelled bisimulation). Upon input of a message* n_a, P *non-deterministically decides whether to diverge or to perform an output on* a *after* n *more steps. On the other hand, upon input of* n_a Q *non-deterministically decides to become either* Q_1 *or* Q_2, *where* Q_1 *performs an output on* a *after* n *steps if* n *is odd and diverges if* n *is even, while* Q_2 *performs an output on* a *after* n *steps if* n *is even and diverges if* n *is odd.*

After the choice of the first process we need to make a choice in the second process, dependent on whether n *is even or odd. Symbolically, in order to make the choice in the second process we need to describe the the condition "*n *is even (odd)" using a disjunction of formulas. We conjecture that this cannot be done with a finite disjunction (of finite formulas) in this guard and expression language.*

$$
\begin{aligned}
P &= a(x).\Omega + a(x).(\nu c)\,(P'(x) \mid !c(y).P'(y)) \\
P'(y) &= \overline{y}\langle a\rangle + \overline{c}\langle \mathtt{p}(y)\rangle \\
Q &= (\nu c)\,((a(x).Q_1(x) + a(x).Q_2(x)) \mid !c(y).Q_2(y)) \\
Q_1(x) &= [\, x : \mathcal{N}\,]\Omega + \overline{c}\langle \mathtt{p}(x)\rangle) \\
Q_2(x) &= \overline{x}\langle a\rangle + (\nu d)\,(\overline{d}\langle \mathtt{p}(x)\rangle \mid d(z).Q_1(z)) \\
\Omega &= (\nu c)\,(\overline{c}\langle c\rangle \mid !c(z).\overline{c}\langle c\rangle)
\end{aligned}
$$

The question then arises if it would be possible to extend the logical language used in environments to always enable a finite decomposition. However, this would need a more powerful logic: A more sophisticated version of this example would use that the (finite-control) spi calculus (with symmetric encryption and pairing) is Turing-complete [Hüt02]. We could then let P receive a Turing machine and its input (in some suitable encoding) and choose between diverging or simulating the machine, signalling failure or success upon termination. Q would make an initial choice and simulate the machine in both cases, diverging on failure (resp. success) and signalling success (resp. failure). In this way, a finite decomposition (in an exogenous logic, i.e., one where the process terms are not part of the logical language) would require representing the predicate
"$t \in \{(M\ .\ N)$ where M codes for a Turing machine that accepts (rejects) $N\}$". This is clearly also an issue for automated verification. However, in our experiments with simple security protocols (described in Appendix B) we have not had use for any decomposition, suggesting that the actual impact of this issue is highly domain-dependent.

4.2.2 Symbolic Bisimulation

A *symbolic relation* $\mathcal{R}$ is a subset of $\mathbf{SE} \times \mathcal{P} \times \mathcal{P}$. We write $se \vdash P\ \mathcal{R}\ Q$ for $(se, P, Q) \in \mathcal{R}$. $\mathcal{R}$ is *symmetric* if whenever $se \vdash P\ \mathcal{R}\ Q$ we have that $(th^{\mathrm{T}}, tv^{\mathrm{T}}, ((\nu D)\,\psi, (\nu C)\,\phi)) \vdash Q\ \mathcal{R}\ P$. $\mathcal{R}$ is *consistent* if se is consistent and $\mathrm{fv}(P, Q) \subseteq \mathrm{dom}(tv)$ whenever $se \vdash P\,\mathcal{R}\,Q$.

Intuitively, for two processes to be bisimilar under a given environment every *possible* and *detectable* transition of one of the processes must be simulated by a transition of the other process on a *corresponding* channel such that the *updated* environment is consistent. The consistency of the updated environment implies that the simulating transition is also possible and detectable.

Definition 4.2.13 *A symmetric consistent symbolic relation $\mathcal{R}$ is a* symbolic bisimulation *if whenever $se \vdash P\,\mathcal{R}\,Q$ with $t = \max(\mathrm{range}(th) \cup \mathrm{range}(tv))$ then*

- *If $P \xrightarrow[\phi']{(\nu\tilde{c})\,\tau}_s P'$ with $\{\tilde{c}\} \cap \mathrm{n}_1(se) = \emptyset$, and there are σ, ρ, B with $se \vdash \sigma \leftrightarrow_B \rho$, $[\![\phi'\sigma]\!]$ and $(\{\tilde{c}\} \cup \mathrm{fn}(P, Q)) \cap B = \emptyset$,*
  then there is a decomposition $\{se_i\}_{i \in I}$ of $(th, tv, ((\nu C \cup \{\tilde{c}\})\,\phi \wedge \phi', (\nu D)\,\psi))$ such that for each $i \in I$, there are $\{\tilde{e}\}, \psi', Q'$ with $Q \xrightarrow[\psi']{(\nu\tilde{e})\,\tau}_s Q'$, $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$ and $(th, tv, ((\nu C \cup \{\tilde{c}\})\,\phi_i, (\nu D \cup \{\tilde{e}\})\,\psi_i)) \vdash tt \leftrightarrow \psi'$. Finally, we require $(th, tv, ((\nu C \cup \{\tilde{c}\})\,\phi_i, (\nu D \cup \{\tilde{e}\})\,\psi_i)) \vdash P'\,\mathcal{R}\,Q'$.
- *If $P \xrightarrow[\phi']{(\nu\tilde{c})\,F(x)}_s P'$ with $\{\tilde{c}\} \cap \mathrm{n}_1(se) = \emptyset$ and $x \notin \mathrm{dom}(tv)$, and there are σ, ρ, B with $se \vdash \sigma \leftrightarrow_B \rho$, $[\![\phi'\sigma]\!]$, $\mathbf{e}(F\sigma) \in \pi_1(\mathbf{C}^B_{\sigma,\rho}(th))$ and $(\{\tilde{c}\} \cup \mathrm{fn}(P, Q)) \cap B = \emptyset$,*

then there are $y \notin (\mathrm{dom}(tv) \cup \{x\})$ *and a decomposition* $\{se_i\}_{i\in I}$ *of* $(th, tv', ((\nu C \cup \{\tilde{c}\})\, \phi \wedge \phi' \wedge [\, y = F\,], (\nu D)\, \psi))$ *where* $tv' = tv \cup \{x \mapsto t{+}1, y \mapsto t{+}1\}$ *such that for each* $i \in I$, *there are* $\{\tilde{e}\}, \psi', Q', F'$ *with* $Q \xrightarrow[\psi']{(\nu \tilde{e})\, F'(x)}_s Q'$, $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$ *and* $(th, tv', ((\nu C \cup \{\tilde{c}\})\, \phi_i, (\nu D \cup \{\tilde{e}\})\, \psi_i)) \vdash tt \leftrightarrow \psi' \wedge [\, y = F'\,]$. *Finally, we require* $(th, tv', ((\nu C \cup \{\tilde{c}\})\, \phi_i, (\nu D \cup \{\tilde{e}\})\, \psi_i)) \vdash P' \mathcal{R}\, Q'$.

- *If* $P \xrightarrow[\phi']{(\nu \tilde{c})\, \overline{F}\, G}_s P'$ *with* $\{\tilde{c}\} \cap \mathrm{n}_1(se) = \emptyset$, *and* *there are* σ, ρ, B *with* $se \vdash \sigma \leftrightarrow_B \rho$, $[\![\phi'\sigma]\!]$, $\mathbf{e}(F\sigma) \in \pi_1(\mathbf{C}^B_{\sigma,\rho}(th))$, $x \notin \mathrm{dom}(tv)$ *and* $(\{\tilde{c}\} \cup \mathrm{fn}(P, Q)) \cap B = \emptyset$, *then there are* $y \notin \mathrm{dom}(tv)$ *and a decomposition* $\{se_i\}_{i\in I}$ *of* $(th, tv', ((\nu C \cup \{\tilde{c}\})\, \phi \wedge \phi' \wedge [\, y = F\,], (\nu D)\, \psi))$ *where* $tv' = tv \cup \{y \mapsto t{+}1\}$ *such that for each* $i \in I$, *there are* $\{\tilde{e}\}, \psi', Q', F', G'$ *with* $Q \xrightarrow[\psi']{(\nu \tilde{e})\, \overline{F'}\, G'}_s Q'$, $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$ *and* $(th', tv', ((\nu C \cup \{\tilde{c}\})\, \phi_i, (\nu D \cup \{\tilde{e}\})\, \psi_i)) \vdash tt \leftrightarrow \psi' \wedge [\, y = F'\,]$ *where* $th' = th \cup \{(G, G') \mapsto i{+}1\}$ *if* $G, G' \notin \mathrm{dom}(th)$, $th' = th$ *otherwise.* *Then* $(th', tv', ((\nu C \cup \{\tilde{c}\})\, \phi_i, (\nu D \cup \{\tilde{e}\})\, \psi_i)) \vdash P' \mathcal{R}\, Q'$.

Symbolic bisimilarity, *written* $\sim_{\mathrm{s}}$, *is the union of all symbolic bisimulations.*

Soundness

Symbolic bisimilarity is sound with respect to concrete bisimilarity.

Lemma 4.2.14 *For all processes* P, Q, *consistent symbolic environments* se, *and substitution pairs* (σ, ρ) *satisfying* $se \vdash \sigma \leftrightarrow_B \rho$ *where* $\mathrm{fn}(P, Q) \cap B = \emptyset$ *we have that*

1. *Any concrete transition (up to* α*-equivalence) of* $P\sigma$ *that must be simulated by* $Q\rho$ *under the concrete environment* $\mathbf{C}^B_{\sigma,\rho}(th)$ *has a corresponding symbolic transition of* P *that must be simulated by* Q *under* se.
2. *If a symbolic transition (up to* α*-equivalence) of* P *is simulated by* Q *under* se, *and has a corresponding concrete transition of* $P\sigma$ *that must be simulated by* $Q\rho$ *under* $\mathbf{C}^B_{\sigma,\rho}(th)$, *then* $Q\rho$ *can simulate the concrete transition.* *Furthermore, there are substitutions* σ', ρ' *such that if* (h'', P'', Q'') *is the resulting hedged process pair,* P' *and* Q' *are the resulting processes after the symbolic step and* se' *is the resulting symbolic environment, then* $se' \vdash \sigma' \leftrightarrow_{B'} \rho'$, $\mathrm{fn}(P', Q') \cap B' = \emptyset$, $h'' = th'(\sigma', \rho')$, $P'\sigma' >_{\mathrm{a}} P''$ *and* $Q'\rho' >_{\mathrm{a}} Q''$. *Here* (σ', ρ') *are dependent on the type of transition:*

- *If it is tau then* $(\sigma', \rho') = (\sigma, \rho)$,
- *If it is an output then* $(\sigma', \rho') = (\sigma\{^{a}/_{y}\}, \rho\{^{b}/_{y}\})$ *where* y *is a fresh variable and* (a, b) *are the channels of the concrete transitions.*
- *If it is an input then* $(\sigma', \rho') = (\sigma\{^{M}/_{x}\}\{^{a}/_{y}\}, \rho\{^{N}/_{x}\}\{^{b}/_{y}\})$ *where* x *is the variable in the symbolic input transitions,* y *is a fresh variable and* (a, b) *are the channels and* (M, N) *are the messages (created by the environment) of the concrete early input transitions.*

$$\begin{array}{ccc} (se, P, Q) & \longrightarrow & (se', P', Q') \\ \Big\downarrow{\scriptstyle \sigma,\rho} & & \Big\downarrow{\scriptstyle \sigma',\rho'} \\ (\mathbf{C}^{B}_{\sigma,\rho}(th), P\sigma, Q\rho) & \longrightarrow & (th'', P'', Q'') \end{array}$$

Proof:

1. We have three cases depending on the type of the transition of $P\sigma$.

   tau If $P\sigma \xrightarrow{\tau} P''$ then by Theorem 4.1.18 there are $\phi', \widetilde{d}, P'$ such that $P \xrightarrow[\phi']{(\nu\tilde{d})\,\tau}_{s} P'$, $[\![\phi'\sigma]\!]$ and $P'\sigma >_{\mathrm{a}} P''$. By Salp we may choose the $\tilde{d}$ such that $\{\tilde{d}\} \cap (\mathrm{n}(se) \cup B) = \emptyset$. Since $[\![\phi'\sigma]\!]$ the transition must be simulated by Q.

   input Assume that $P\sigma \xrightarrow{a(x)} P''$ and $(a, b) \in \mathbf{C}^{B}_{\sigma,\rho}(th)$. We choose a transition α-equivalent to this one where $x \notin \mathrm{dom}(tv)$.
   By Theorem 4.1.18 there are $\phi', P', F, \tilde{d}$ such that $a = \mathbf{e}(F\sigma)$, $P \xrightarrow[\phi']{(\nu\tilde{d})\,F(x)}_{s} P'$ and $[\![\phi'\sigma]\!]$. By Salp we may choose the $\tilde{d}$ such that $\{\tilde{d}\} \cap (\mathrm{n}(se) \cup B) = \emptyset$. Since $[\![\phi'\sigma]\!]$ and $a \in \pi_1(\mathbf{C}^{B}_{\sigma,\rho}(th))$ the transition must be simulated by Q.

   output If $P\sigma \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P''$ and $(a, b) \in \mathbf{C}^{B}_{\sigma,\rho}(th)$ we choose a transition α-equivalent to this one where $\{\tilde{b}\} \cap (\mathrm{n}(se) \cup B) = \emptyset$. By Theorem 4.1.18 there are $\phi', P', \tilde{d}, F, G$ such that $(\nu\tilde{b})\,\overline{a}\,M = \mathbf{e}((\nu\tilde{d})\,\overline{F}\,G)$, $P \xrightarrow[\phi']{(\nu\tilde{d})\,\overline{F}\,G}_{s} P'$ and $[\![\phi'\sigma]\!]$. By Salp we may choose the $\tilde{d}$ such that $\{\tilde{d}\} \cap (\mathrm{n}(se) \cup B) = \emptyset$. Since $[\![\phi'\sigma]\!]$ and $a \in \pi_1(\mathbf{C}^{B}_{\sigma,\rho}(th))$ the transition must be simulated by Q.

2. We have three cases depending on the type of the transition of P.

   tau Assume that $P \xrightarrow[\phi']{(\nu\tilde{d})\,\tau}_{s} P'$, $\{\tilde{d}\} \cap (\mathrm{n}_1(se) \cup B) = \emptyset$, $[\![\phi'\sigma]\!]$ and that $\{se_i\}_{i\in I}$ is a decomposition of $(th, tv, ((\nu C \cup \{\tilde{c}\})\,\phi \wedge \phi', (\nu D)\,\psi))$ such that

$se_i \vdash \sigma \leftrightarrow_B \rho$, $Q \xrightarrow[\psi']{(\nu\tilde{e})\,\tau}_s Q''$ and $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$. Letting $se' = (th, tv, ((\nu C \cup \{\tilde{c}\})\,\phi_i, (\nu D \cup \{\tilde{e}\})\,\psi_i))$, assume that $se' \vdash tt \leftrightarrow \psi'$ and $se' \vdash P' \mathcal{R} Q'$.
Since $\mathrm{fn}(P') \subseteq (\mathrm{fn}(P) \cup \{\tilde{d}\})$ and $\mathrm{fn}(Q') \subseteq (\mathrm{fn}(Q) \cup \{\tilde{e}\})$ we get $\mathrm{fn}(P', Q') \cap B = \emptyset$.
By Theorem 4.1.18 $P\sigma \xrightarrow{\tau} P''$ where $P'\sigma >_\mathrm{a} P''$ and $Q\rho \xrightarrow{\tau} Q''$ where $Q'\rho >_\mathrm{a} Q''$. By assumption $\mathbf{C}^B_{\sigma,\rho}(th)$ is consistent.

input Assume that $P \xrightarrow[\phi']{(\nu\tilde{d})\,F(x)}_s P'$, $\{\tilde{d}\} \cap (\mathrm{n}_1(se) \cup B) = \emptyset$,
$x, y \notin \mathrm{dom}(tv)$, $[\![\phi'\sigma]\!]$, $\mathbf{e}(F\sigma) = a \in \pi_1(\mathbf{C}^B_{\sigma,\rho}(th))$. By the consistency of $\mathbf{C}^B_{\sigma,\rho}(th)$ there is b such that $(a, b) \in \mathbf{C}^B_{\sigma,\rho}(th)$. By Lemma A.1.2(1) we get $G = b$ whenever $(a, G) \in \mathcal{S}(h)$ with h consistent and $(a, b) \in h$.
Assume that $\{se_i\}_{i\in I}$ is a decomposition of
$(th, tv', ((\nu C \cup \{\tilde{c}\})\,\phi \wedge \phi' \wedge [\,y = F\,], (\nu D)\,\psi))$ where
$tv' = tv \cup \{x \mapsto t{+}1, y \mapsto t{+}1\}$.
Take i, M, N, B' such that $se_i \vdash \sigma' \leftrightarrow_{B\cup B'} \rho'$ and $Q \xrightarrow[\psi']{(\nu\tilde{e})\,F'(x)}_s Q'$ where $\sigma' = \sigma\{{}^{M}\!/_{x}\}\{{}^{a}\!/_{y}\}$, $\rho' = \rho\{{}^{N}\!/_{x}\}\{{}^{b}\!/_{y}\}$, $b = \mathbf{e}(F'\rho)$, $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$.
Let $se' = (th, tv', ((\nu C \cup \{\tilde{c}\})\,\phi_i, (\nu D \cup \{\tilde{e}\})\,\psi_i))$ and assume that $se' \vdash tt \leftrightarrow \psi' \wedge [\,y = F'\,]$. Since $x, y \notin \mathrm{v}(\psi')$ we have $[\![\psi'\rho]\!]$. Since $se' \vdash \sigma' \leftrightarrow_{B\cup B'} \rho'$ we get
$(M, N) \in \mathcal{S}(\mathcal{I}(\{(\mathbf{e}(F''\sigma), \mathbf{e}(G''\rho)) \mid (F'', G'') \in \mathrm{dom}(th)\} \cup \mathrm{Id}_{B\cup B'}))$. Since $\mathrm{fn}(P') \subseteq \mathrm{fn}(P)$ and $\mathrm{fn}(Q') \subseteq \mathrm{fn}(Q)$ we also have that $\mathrm{fn}(P', Q') \cap (B \cup B') = \emptyset$.
By Theorem 4.1.18 $P\sigma \xrightarrow{a(x)} P''$ where $P'\sigma >_\mathrm{a} P''$ and $Q\rho \xrightarrow{b(x)} Q''$ where $Q'\rho >_\mathrm{a} Q''$. Then $P'\sigma' >_\mathrm{a} P''\{{}^{M}\!/_{x}\}$ and $Q'\rho' >_\mathrm{a} Q''\{{}^{N}\!/_{x}\}$. By Lemma 3.3.22) $\mathbf{C}^{B\cup B'}_{\sigma',\rho'}(th) = \mathbf{C}^B_{\sigma,\rho}(th) \cup B'$ is consistent.

output Assume that $P \xrightarrow[\phi']{(\nu\tilde{d})\,\overline{F}\,G}_s P'$, $\{\tilde{d}\} \cap (\mathrm{n}_1(se) \cup B) = \emptyset$, $y \notin \mathrm{dom}(tv)$, $[\![\phi'\sigma]\!]$, $\mathbf{e}(F\sigma) \in \pi(\mathbf{C}^B_{\sigma,\rho}(th))$.
By the consistency of $\mathbf{C}^B_{\sigma,\rho}(th)$ there is b such that $(a, b) \in \mathbf{C}^B_{\sigma,\rho}(th)$. By Lemma A.1.2(1) we get $N = b$ whenever $(a, N) \in \mathcal{S}(h)$ with h consistent and $(a, b) \in h$.
Assume that $\{se_i\}_{i\in I}$ is a decomposition of
$(th \cup th', tv', ((\nu C \cup \{\tilde{c}\})\,\phi \wedge \phi' \wedge [\,y = F\,], (\nu D)\,\psi))$ where
$th' := \{(G, G') \mapsto i{+}1 \mid (G, G') \notin \mathrm{dom}(th)\}$ and
$tv' = tv \cup \{y \mapsto t{+}1\}$, and let $\sigma' = \sigma\{{}^{a}\!/_{y}\}$ and $\rho' = \rho\{{}^{b}\!/_{y}\}$. By assumption $Q \xrightarrow[\psi']{(\nu\tilde{e})\,\overline{F'}\,G'}_s Q''$ with $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$ and

$b = \mathbf{e}(F'\rho)$.
We let $se' = (th \cup th', tv', ((\nu C \cup \{\tilde{c}\})\,\phi_i, (\nu D \cup \{\tilde{e}\})\,\psi_i))$. By assumption $se' \vdash tt \leftrightarrow \psi' \wedge [\,y = F'\,]$.
Because of the name freshness conditions and since we only add information to the environment, $se' \vdash \sigma' \leftrightarrow_B \rho'$, so since $y \notin \mathrm{v}(\phi')$ we have $[\![\psi'\rho]\!]$.
By Theorem 4.1.18 there are $\widetilde{b}, \widetilde{c}, P'', Q'', a, b, M, N$ such that $\{\tilde{b}\} \subseteq \{\tilde{e}\}$, $\{\tilde{c}\} \subseteq \{\tilde{e}\}$ and $P\sigma \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P''$, where $\mathbf{e}(F\sigma) = a$, $\mathbf{e}(G\sigma) = M$ and $P'\sigma >_{\mathrm{a}} P''$, and $Q\rho \xrightarrow{(\nu\tilde{c})\,\overline{b}\,N} Q''$, where $\mathbf{e}(F'\rho) = b$, $\mathbf{e}(G'\rho) = N$ and $Q'\rho >_{\mathrm{a}} Q''$. $\mathbf{C}^B_{\sigma',\rho'}(th \cup th')$ is consistent by the consistency of se'. There are two cases for th'.
If $th' = \emptyset$, i.e., $(G, G') \in \mathrm{dom}(th)$, then $(M, N) \in \mathcal{S}(\mathbf{C}^B_{\sigma,\rho}(th))$, so $\mathbf{C}^B_{\sigma',\rho'}(th \cup th') = \mathbf{C}^B_{\sigma,\rho}(th) = \mathcal{I}(\mathbf{C}^B_{\sigma,\rho}(th) \cup \{(M, N)\})$. Otherwise, $th' = \{(G, G') \mapsto i{+}1\}$ and using Lemma 3.3.12(5)

$$
\begin{aligned}
&\mathbf{C}^B_{\sigma',\rho'}(th \cup th') \\
&\quad = \mathcal{I}(\{(\mathbf{e}(F\sigma), \mathbf{e}(G\rho)) \mid (F, G) \in \mathrm{dom}(th)\} \cup \{(M, N')\} \cup \mathrm{Id}_B) \\
&\quad = \mathcal{I}(\mathcal{I}(\{(\mathbf{e}(F\sigma), \mathbf{e}(G\rho)) \mid (F, G) \in \mathrm{dom}(th)\} \cup \mathrm{Id}_B) \cup \{(M, N')\}) \\
&\quad = \mathcal{I}(\mathbf{C}^B_{\sigma,\rho}(th) \cup \{(M, N)\}).
\end{aligned}
$$

□

Since we are reasoning up to bijective renaming and labelled bisimilarity, we need to show the soundness of these up-to techniques for concrete hedged bisimulation.

Lemma 4.2.15 *Hedged bisimulation is sound up to bijective renaming and labelled bisimulation.*

Proof: Let $\mathcal{R}$ be a hedged bisimulation up to bijective renaming and labelled bisimulation. We wish to show that $\mathcal{R} \subseteq \sim_{\mathrm{h}}$. We do this by showing that $\mathcal{R}_{\mathrm{bl}}$ is a hedged bisimulation; the result follows since $\mathcal{R} \subseteq \mathcal{R}_{\mathrm{bl}}$ because "up to bijective renaming and labelled bisimulation" is an expansion. The proof is standard, and present in Section A.2. □

Finally, we have the soundness theorem.

Theorem 4.2.16 *For all processes P, Q, and symbolic environments se such that $se \vdash P \sim_{\mathrm{s}} Q$ we have that*
$\mathbf{C}^B_{\sigma,\rho}(se) \vdash P\sigma \sim_{\mathrm{h}} Q\rho$ for all $B \subset \mathcal{N}$ with $\mathrm{fn}(P, Q) \cap B = \emptyset$ and substitution pairs (σ, ρ) satisfying $se \vdash \sigma \leftrightarrow_B \rho$.

PROOF: By Lemma 4.2.14 and Lemma 4.1.7,

$$\mathcal{R} := \{(\mathbf{C}^B_{\sigma,\rho}(th), P\sigma, Q\rho) \mid se' \vdash P \sim_{\mathrm{s}} Q \text{ and } \exists B : se' \vdash \sigma \leftrightarrow_B \rho\}$$

is a hedged bisimulation up to bijective renaming and labelled bisimilarity. By Lemma 4.2.15, $\mathcal{R} \subseteq \sim_{\mathrm{h}}$. □

Completeness

We show that a symbolic bisimulation that always fully decomposes environments is complete with respect to hedged bisimilarity.

Theorem 4.2.17 *Assume that se, P, Q are such that se is consistent and $\mathbf{C}^B_{\sigma,\rho}(se) \vdash P\sigma \sim_{\mathrm{h}} Q\rho$ whenever $se \vdash \sigma \leftrightarrow_B \rho$ with $B \cap \mathrm{fn}(P, Q) = \emptyset$. Then $se \vdash P \sim_{\mathrm{s}} Q$.*

PROOF: We prove that the set $\mathcal{R} = \{(se, P, Q) \mid se$ is consistent and $\mathbf{C}^B_{\sigma,\rho}(se) \vdash P\sigma \sim_{\mathrm{h}} Q\rho$ whenever $se \vdash \sigma \leftrightarrow_B \rho$ with $B \cap \mathrm{fn}(P, Q) = \emptyset\}$ is a symbolic bisimulation. $\mathcal{R}$ is symmetric, by the symmetry of $\sim_{\mathrm{h}}$ and the definitions of consistency, $se \vdash \sigma \leftrightarrow_B \rho$ and $\mathbf{C}^B_{\sigma,\rho}(se)$. Assume that $se \vdash P \mathcal{R} Q$, $P \xrightarrow[\phi']{\mu_s}_s P'$ and $t = \max(\mathrm{range}(th) \cup \mathrm{range}(tv))$. We consider three cases for μ_s.

tau Assume that $P \xrightarrow[\phi']{(\nu\tilde{c})\,\tau}_s P'$ and $se \vdash \sigma \leftrightarrow_B \rho$ with $\{\tilde{c}\} \cap (\mathrm{n}_1(se) \cup B) = \emptyset$ and $[\![\phi'\sigma]\!]$. We let $\{se_{(\sigma,\rho,B)}\}_{(\sigma,\rho,B)\in I}$ be the full decomposition of $(th, tv, ((\nu C \cup \{\tilde{c}\})\,\phi \wedge \phi', (\nu D)\,\psi))$. Take (σ, ρ, B) with $B \cap \mathrm{fn}(P, Q) = \emptyset$. Note that $se \vdash \sigma \leftrightarrow_B \rho$ and $\mathbf{C}^B_{\sigma,\rho}(se) = \mathbf{C}^B_{\sigma,\rho}(se_{(\sigma,\rho,B)})$. Then $\mathbf{C}^B_{\sigma,\rho}(se_{(\sigma,\rho,B)}) \vdash P\sigma \sim_{\mathrm{h}} Q\rho$.
By Theorem 4.1.18 $P\sigma \xrightarrow{\tau} P''$ where $P'\sigma >_{\mathrm{a}} P''$.
By bisimilarity $Q\rho \xrightarrow{\tau} Q''$ where $\mathbf{C}^B_{\sigma,\rho}(se) \vdash P'' \sim_{\mathrm{h}} Q''$.
By Theorem 4.1.18 $Q \xrightarrow[\psi']{(\nu\tilde{e})\,\tau}_s Q'$ such that $[\![\psi'\rho]\!]$ and $Q'\rho >_{\mathrm{a}} Q''$. Using SALP we may choose $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$.
Since $>_{\mathrm{a}}$ is a labelled bisimulation (Lemma 4.1.7) and $\sim_{\mathrm{h}}$ is sound up to labelled bisimulation (Lemma 4.2.15), $\mathbf{C}^B_{\sigma,\rho}(se_{(\sigma,\rho,B)}) \vdash P'\sigma \sim_{\mathrm{h}} Q'\rho$. Thus $\mathbf{C}^B_{\sigma,\rho}(se_{(\sigma,\rho,B)})$ is consistent.
Since $\mathrm{fn}(P') \subseteq \mathrm{fn}(P)$ and $\mathrm{fn}(Q') \subseteq \mathrm{fn}(Q)$ we get $\mathrm{fn}(P', Q') \cap B = \emptyset$.
Let $se' = (th, tv, ((\nu C \cup \{\tilde{c}\})\,\phi_{(\sigma,\rho,B)}, (\nu D \cup \{\tilde{e}\})\,\psi_{(\sigma,\rho,B)} \wedge \psi'))$. Then se' has the unique solution (σ, ρ, B), so $se' \vdash P' \mathcal{R} Q'$.

output Assume that $P \xrightarrow[\phi']{(\nu\tilde{b})\,\overline{F}\,G}_s P'$, $se \vdash \sigma \leftrightarrow_B \rho$, $\{\tilde{b}\} \cap (\mathrm{n}_1(se) \cup B) = \emptyset$, $[\![\phi'\sigma]\!]$ and $y \notin \mathrm{dom}(tv)$.

We let $tv' = tv \cup \{y \mapsto t{+}1\}$ and $th' = th \cup \{(G, G') \mapsto i{+}1\}$ if $G, G' \notin \mathrm{dom}(th)$, $th' = th$ otherwise. We then let $\{se_{(\sigma,\rho,B)}\}_{(\sigma,\rho,B)\in I}$ be the full decomposition of $(th, tv', ((\nu C \cup \{\tilde{c}\})\, \phi \wedge \phi' \wedge [\, y = F\,], (\nu D)\, \psi))$.

Take any $(\sigma, \rho, B) \in I$ and let $\sigma = \sigma'\{^{F}/_{y}\}$ and $\rho = \rho'\{^{G}/_{y}\}$. Note that $F = a$ and $G = b$ with $se \vdash \sigma' \leftrightarrow_B \rho'$, $(a, b) \in \mathbf{C}^B_{\sigma',\rho'}(se)$ and $\mathbf{C}^B_{\sigma',\rho'}(se) = \mathbf{C}^B_{\sigma,\rho}(se_{(\sigma,\rho,B)})$. Then $\mathbf{C}^B_{\sigma,\rho}(se_{(\sigma,\rho,B)}) \vdash P\sigma \sim_{\mathrm{h}} Q\rho$.

By Theorem 4.1.18 $P\sigma \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} P''$ where $a = \mathbf{e}(F\sigma)$, $M = \mathbf{e}(G\sigma)$, $\{\tilde{c}\} = \{\tilde{b}\} \cap \mathrm{en}(\mathbf{e}_{\mathrm{a}}(G))$ and $P'\sigma >_{\mathrm{a}} P''$. By bisimilarity $Q\rho \xrightarrow{(\nu\tilde{d})\,\overline{b}\,N} Q''$ where $\mathbf{C}^B_{\sigma,\rho}(se) \vdash a \leftrightarrow b$ and $\mathcal{I}(\mathbf{C}^B_{\sigma,\rho}(se) \cup \{(M, N)\}) \vdash P'' \sim_{\mathrm{h}} Q''$. Using ALP we may choose $\{\tilde{d}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$.

By Theorem 4.1.18 $Q \xrightarrow[\psi']{(\nu\tilde{e})\,\overline{F'}\,G'}_s Q'$ such that $b = \mathbf{e}(F'\rho)$, $N = \mathbf{e}(G'\rho)$, $\{\tilde{d}\} = \{\tilde{e}\} \cap \mathrm{en}(\mathbf{e}_{\mathrm{a}}(G))$, $[\![\psi'\rho]\!]$ and $Q'\rho >_{\mathrm{a}} Q''$. Using SALP we may choose $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$.

Since $\sim_{\mathrm{h}}$ is sound up to $>_{\mathrm{a}}$ (Lemma 4.1.7, Lemma 4.2.15), $\mathcal{I}(\mathbf{C}^B_{\sigma,\rho}(se) \cup \{(M, N)\}) \vdash P\sigma \sim_{\mathrm{h}} Q\rho$. Thus $\mathcal{I}(\mathbf{C}^B_{\sigma,\rho}(se) \cup \{(M, N)\})$ is consistent. Since $\mathrm{fn}(P') \subseteq (\mathrm{fn}(P) \cup \{\tilde{c}\})$ and $\mathrm{fn}(Q') \subseteq (\mathrm{fn}(Q) \cup \{\tilde{d}\})$ we get $\mathrm{fn}(P', Q') \cap B = \emptyset$. Then

$se' = (th \cup th', tv, ((\nu C \cup \{\tilde{c}\})\, \phi_{(\sigma,\rho,B)} (\nu D \cup \{\tilde{e}\})\, \psi_{(\sigma,\rho,B)}))$ has the unique solution (σ, ρ, B) where $\mathbf{C}^B_{\sigma',\rho'}(se') = \mathcal{I}(\mathbf{C}^B_{\sigma,\rho}(se) \cup \{(M, N)\})$, so $se' \vdash P' \,\mathcal{R}\, Q'$. Moreover, $se' \vdash tt \leftrightarrow \psi' \wedge [\, y = G\,]$.

input Assume that $P \xrightarrow[\phi']{(\nu\tilde{c})\,F(x)}_s P'$ and that $se \vdash \sigma \leftrightarrow_B \rho$ with $\{\tilde{b}\} \cap (\mathrm{n}_1(se) \cup B) = \emptyset$, $a = \mathbf{e}(F\sigma)$, $\mathbf{C}^B_{\sigma,\rho}(se) \vdash a \leftrightarrow b$ and $[\![\phi'\sigma]\!]$ and that $x, y \notin \mathrm{dom}(tv)$.

We let $\{se_{(\sigma,\rho,B)}\}_{(\sigma,\rho,B)\in I}$ be the full decomposition of $\{(th, tv \cup \{y \mapsto t{+}1\}, ((\nu C \cup \{\tilde{c}\})\, \phi \wedge \phi' \wedge [\, y = F\,], (\nu D)\, \psi))\}$. Take $(\sigma, \rho, B) \in I$. We let $J_{(\sigma,\rho,B)} = \{(B', M, N)$ with $B' \cap (B \cup \{\tilde{b}\} \cup \mathrm{n}(se) \cup \mathrm{n}(P, Q)) = \emptyset$, $\mathbf{C}^B_{\sigma,\rho}(se) \cup \mathrm{Id}_{B'} \vdash M \leftrightarrow N$ and $B' \subseteq \mathrm{n}(M, N)\}$. We also let $tv' = tv \cup \{x \mapsto t{+}1, y \mapsto t{+}2\}$, $\phi_M = \phi_{(\sigma,\rho,B)} \wedge [\, x = M\,]$ and $\psi_N = \psi(\sigma, \rho, B) \wedge [\, x = N\,]$. Then

$\{(th, tv', ((\nu C \cup \{\tilde{c}\})\, \phi_M, (\nu D)\, \psi_N)) \mid (\sigma, \rho, B) \in I \wedge (B', M, N) \in J_{(\sigma,\rho,B)}\}$ is a consistent decomposition of $\{(th, tv', ((\nu C \cup \{\tilde{c}\})\, \phi \wedge \phi' \wedge [\, y = F\,], (\nu D)\, \psi))\}$. Take any $(B', M, N) \in J_{(\sigma,\rho,B)}$.

By Theorem 4.1.18 $P\sigma \xrightarrow{a(x)} P''$ where $P'\sigma >_{\mathrm{a}} P''$. By bisimilarity $Q\rho \xrightarrow{b(x)} Q''$ where $\mathbf{C}^B_{\sigma,\rho}(se) \cup \mathrm{Id}_{B'} \vdash P''\{^{M}/_{x}\} \sim_{\mathrm{h}} Q''\{^{N}/_{x}\}$.

By Theorem 4.1.18 $Q \xrightarrow[\psi']{(\nu\tilde{e})\,F'(x)}_s Q'$ such that $[\![\psi'\rho]\!]$ and $Q'\rho >_{\mathrm{a}} Q''$. Using SALP we may choose $\{\tilde{e}\} \cap (\mathrm{n}_2(se) \cup B) = \emptyset$.

Since $\sim_{\mathrm{h}}$ is sound up to $>_{\mathrm{a}}$ (Lemma 4.1.7, Lemma 4.2.15), $\mathbf{C}^{B}_{\sigma,\rho}(se) \cup B' \vdash P\sigma\{^{M}/_{x}\} \sim_{\mathrm{h}} Q\rho\{^{N}/_{x}\}$. Since $\mathrm{fn}(P') \subseteq (\mathrm{fn}(P) \cup B')$ and $\mathrm{fn}(Q') \subseteq (\mathrm{fn}(Q) \cup B')$ we get $\mathrm{fn}(P', Q') \cap B = \emptyset$.
Then $se' = (th, tv', ((\nu C \cup \{\tilde{c}\})\,\phi_M, (\nu D \cup \{\tilde{e}\})\,\psi_N))$ has the unique solution $(\sigma', \rho', B \cup B')$ where $\sigma' = \sigma\{^{M}/_{x}\}\{^{a}/_{y}\}$ and $\rho' = \rho\{^{N}/_{x}\}\{^{b}/_{y}\}$ since $\mathbf{C}^{B\cup B'}_{\sigma',\rho'}(se') = \mathbf{C}^{B}_{\sigma,\rho}(se) \cup \mathrm{Id}_{B'}$ is consistent, so $se' \vdash P' \;\mathcal{R}\; Q'$. Moreover, $se' \vdash tt \leftrightarrow \psi' \wedge [\,y = F'\,]$.

□

4.3 Examples

4.3.1 Potential Sources of Incompleteness

The processes in the following examples are taken from [BBN04], where they were given as examples of the incompleteness of the earlier version of symbolic bisimilarity (lacking distinctions) proposed in that paper. All these examples start from the same symbolic environment $se := (\{(a,a) \mapsto 0\}, \emptyset, (tt, tt))$. Since se has no variables, it has the unique solution $h := \mathbf{C}^{\emptyset}_{\epsilon,\epsilon}(\{(a,a) \mapsto 0\}) = \{(a,a)\}$. We assume that x, y, z, a, k, n are pair-wise different wherever they occur below.

Guard Decomposition

The first example shows how decompositions permit a simple case split.

Example 4.3.1 *Let*

$$\begin{array}{rcll} P_1 & := & a(x).\overline{a}\langle a\rangle & \\ Q_1 & := & a(x).Q_1' & Q_1' := ([\,x = a\,]\overline{a}\langle a\rangle + \neg[\,x = a\,]\overline{a}\langle a\rangle). \end{array}$$

Then $se \vdash P_1 \sim_{\mathrm{s}} Q_1$. *Specifically, the symmetric closure of the set*

$$\mathcal{R} \;:=\; \{(se, P_1, Q_1), (se_1, \overline{a}\,a, Q_1'), (se_2, \mathbf{0}, \mathbf{0}), (se_3, \mathbf{0}, \mathbf{0}) \mid x, y, z \in \mathcal{V}\}$$

where

$$\begin{array}{rcl} se_1 & := & (\{(a,a) \mapsto 0\}, \{x \mapsto 1, y \mapsto 1\}, ([\,y = a\,], [\,y = a\,])) \\ se_2 & := & (\{(a,a) \mapsto 0\}, \{x \mapsto 1, y \mapsto 1, z \mapsto 2\}, ([\,x = a\,] \wedge [\,y = a\,] \wedge [\,z = a\,], \\ & & \quad [\,x = a\,] \wedge [\,y = a\,] \wedge [\,z = a\,])) \\ se_3 & := & (\{(a,a) \mapsto 0\}, \{x \mapsto 1, y \mapsto 1, z \mapsto 2\}, ((\neg[\,x = a\,]) \wedge [\,y = a\,] \wedge [\,z = a\,], \\ & & \quad (\neg[\,x = a\,]) \wedge [\,y = a\,] \wedge [\,z = a\,])) \end{array}$$

is a symbolic bisimulation. We consider $(se_1, \overline{a}\,a, Q_1')$. *The symbolic transition* $P \xrightarrow[[\,a:\mathcal{N}\,]\wedge[\,a:\mathcal{M}\,]]{\overline{a}\langle a\rangle} \mathbf{0}$ *is possible and detectable: Letting* $\sigma = \{{}^{a}/_{x}\}\{{}^{a}/_{y}\}$ *we have* $se_1 \vdash \sigma \leftrightarrow_{\emptyset} \sigma$, $a \in \pi_1(\mathbf{C}^{\emptyset}_{\sigma,\sigma}(se_1)) = \{a\}$ *and* $[\![([\,a:\mathcal{N}\,] \wedge [\,a:\mathcal{M}\,])\sigma]\!]$.

We choose $\{se_2, se_3\}$ *as a decomposition of* $(\{(a,a) \mapsto 0\}, \{x \mapsto 1, y \mapsto 1, z \mapsto 2\}, ([\,y = a\,] \wedge [\,a:\mathcal{N}\,] \wedge [\,a:\mathcal{M}\,] \wedge [\,z = a\,], [\,y = a\,]))$*:* se_2 *and* se_3 *are both consistent since they are symmetric, and for all* $\rho : \{x, y, z\} \to \mathcal{M}$ *we have either* $[\![[\,x = a\,]\rho]\!]$ *or* $[\![\neg[\,x = a\,]\rho]\!]$.

Considering se_2, $Q_2' \xrightarrow[[\,a:\mathcal{N}\,]\wedge[\,a:\mathcal{M}\,]\wedge[\,x=a\,]]{\overline{a}\langle a\rangle} \mathbf{0}$ *where trivially* $se_2 \vdash tt \leftrightarrow [\,a:\mathcal{N}\,] \wedge [\,a:\mathcal{M}\,] \wedge [\,x = a\,] \wedge [\,z = a\,]$.

Similarly, $Q_2' \xrightarrow[[\,a:\mathcal{N}\,]\wedge[\,a:\mathcal{M}\,]\wedge\neg[\,x=a\,]]{\overline{a}\langle a\rangle} \mathbf{0}$ *with* $se_3 \vdash tt \leftrightarrow [\,a:\mathcal{N}\,] \wedge [\,a:\mathcal{M}\,] \wedge (\neg[\,x = a\,]) \wedge [\,z = a\,]$.

Delayed instantiation

In general, symbolic bisimulations let us postpone the "instantiation" of input variables until the moment they are actually used, leading to a stronger relation.

Example 4.3.2 *Let*

$$\begin{array}{rlrl} P_2 := & a(x).P_2' & P_2' := & (\nu c)\,(\overline{c}\langle c\rangle \mid c(z) \mid c(z).[\,x = a\,]\overline{a}\langle a\rangle) \\ Q_2 := & a(x).Q_2' & Q_2' := & (\nu c)\,(\overline{c}\langle c\rangle \mid c(z) \mid [\,x = a\,]c(z).\overline{a}\langle a\rangle). \end{array}$$

Then $se \vdash P_2 \sim_{\mathrm{s}} Q_2$. *Similarly to before, the symmetric closure of the set*

$$\begin{array}{rcl} \mathcal{R} & := & \{(se, P_1, Q_1)\} \\ & & \cup\{(se_1, \overline{a}\,a, Q_1') \mid x, y \in \mathcal{V}\} \\ & & \cup\{(se_2, \mathbf{0}, \mathbf{0} \mid \neg[\,x = a\,]\overline{a}\langle a\rangle) \mid x, y, z \in \mathcal{V}\} \\ & & \cup\{(se_3, \mathbf{0}, [\,x = a\,]\overline{a}\langle a\rangle \mid \mathbf{0}) \mid x, y, z \in \mathcal{V}\} \end{array}$$

where

$$\begin{array}{rcl} se_1 & := & (\{(a,a) \mapsto 0\}, \{x \mapsto 1, y \mapsto 2\}, ([\,y = a\,], [\,y = a\,])) \\ se_2 & := & (\{(a,a) \mapsto 0\}, \{x \mapsto 1, y \mapsto 2, z \mapsto 3\}, ([\,x = a\,] \wedge [\,y = a\,] \wedge [\,z = a\,], \\ & & \quad [\,x = a\,] \wedge [\,y = a\,] \wedge [\,z = a\,])) \\ se_3 & := & (\{(a,a) \mapsto 0\}, \{x \mapsto 1, y \mapsto 2, z \mapsto 3\}, ((\neg[\,x = a\,]) \wedge [\,y = a\,] \wedge [\,z = a\,], \\ & & \quad (\neg[\,x = a\,]) \wedge [\,y = a\,] \wedge [\,z = a\,])) \end{array}$$

is a symbolic bisimulation.

Detecting Process Actions

Orthogonally to the possibility to decompose, we have added the condition that the environment can detect the process action.

Example 4.3.3 *Let*

$$\begin{aligned} P_3 &:= a(x).(\nu k)\,\overline{a}\langle \mathrm{E}_k(x)\rangle.(\nu n)\,\overline{a}\langle \mathrm{E}_{\mathrm{E}_k(a)}(n)\rangle.\overline{n}\,a \\ Q_3 &:= a(x).(\nu k)\,\overline{a}\langle \mathrm{E}_k(x)\rangle.(\nu n)\,\overline{a}\langle \mathrm{E}_{\mathrm{E}_k(a)}(n)\rangle.[\,x = a\,]\overline{n}\,a. \end{aligned}$$

Then $se \vdash P_3 \sim_{\mathrm{s}} Q_3$*: After the first three transitions we have the symbolically hedged process pair* $(se', \overline{n}\,a, [\,x = a\,]\overline{n}\,a)$ *where*

$$\begin{aligned} se' &:= (th', tv', ((\nu\{k\})\,\phi', (\nu\{k\})\,\psi') \\ th' &:= (\{(a,a) \mapsto 0, (\mathrm{E}_k(x), \mathrm{E}_k(x)) \mapsto 2, (\mathrm{E}_{\mathrm{E}_k(a)}(n), \mathrm{E}_{\mathrm{E}_k(a)}(n)) \mapsto 3\} \\ tv' &:= \{x \mapsto 1, y_1 \mapsto 1, y_2 \mapsto 2, y_3 \mapsto 3\} \\ \phi' &:= [\,y_1 = a\,] \wedge [\,y_2 = a\,] \wedge [\,y_3 = a\,] \\ \psi' &:= \phi' \end{aligned}$$

The symbolic transitions of $\overline{n}\,a$ *and* $[\,x = a\,]\overline{n}\,a$ *are*

$$\overline{n}\,a \xrightarrow[[\,n:\mathcal{N}\,]\wedge[\,a:\mathcal{M}\,]]{\overline{n}\,a}_{s} \mathbf{0} \qquad\qquad [\,x = a\,]\overline{n}\,a \xrightarrow[[\,n:\mathcal{N}\,]\wedge[\,a:\mathcal{M}\,]\wedge[\,x=a\,]]{\overline{n}\,a}_{s} \mathbf{0}$$

Let $\sigma := \{^{a}/_{x}\}$*. As* $se' \vdash \sigma \leftrightarrow_{\emptyset} \sigma$ *and* $\mathbf{C}^{\emptyset}_{\sigma,\sigma}(th') = \{(a,a), (\mathrm{E}_k(a), \mathrm{E}_k(a)), (n,n)\}$*, we have that* $n \in \pi_1(\mathbf{C}^{\emptyset}_{\sigma,\sigma}(th'))$*, so the transition of* $\overline{n}\,a$ *must be simulated by* $[\,x = a\,]\overline{n}\,a$*. The environment after the step is* $se'' := (th', tv' \cup \{z \mapsto 5\}, ((\nu\{k,n\})\,\phi' \wedge [\,z = n\,], (\nu\{k,n\})\,\phi' \wedge [\,z = n\,]))$*.*

We need to show that $se'' \vdash tt \leftrightarrow [\,n:\mathcal{N}\,] \wedge [\,a:\mathcal{M}\,] \wedge [\,x = a\,]$*, i.e., that* $\rho'(x) = a$ *whenever* $se'' \vdash \sigma' \leftrightarrow_B \rho'$*. First note that* $[\![(\phi' \wedge [\,z = n\,])\rho]\!]$ *iff* $a = \rho(y_1) = \rho(y_2) = \rho(y_3)$ *and* $\rho(z) = n$*; we let* $\rho = \{^{a}/_{y_1}\}\{^{a}/_{y_2}\}\{^{a}/_{y_3}\}\{^{n}/_{z}\}$*.*

Assume that $\sigma' = \{^{M}/_{x}\}$ *and* $\rho' = \{^{N}/_{x}\}$ *such that* $se'' \vdash \rho\sigma' \leftrightarrow_B \rho\rho'$*. We let* $h' = \{(a,a), (\mathrm{E}_k(M), \mathrm{E}_k(N)), (\mathrm{E}_{\mathrm{E}_k(a)}(n), \mathrm{E}_{\mathrm{E}_k(a)}(n))\}$*. In order to have* $\rho(z) = n$ *we must have* $(n,n) \in \mathcal{S}(\mathbf{C}^{B}_{\rho\sigma',\rho\rho'}(th')) = \mathcal{S}(\mathcal{I}(h' \cup \mathrm{Id}_B))$*. Since* $\{k,n\}$ *is restricted we cannot have* $k, n \in B$*.*

Then the only way to derive $(n,n) \in \mathcal{A}(h' \cup \mathrm{Id}_B)$ *is by generating* $(\mathrm{E}_k(a), \mathrm{E}_k(a)) \in \mathcal{SA}(h' \cup \mathrm{Id}_B)$ *to analyze* $(\mathrm{E}_{\mathrm{E}_k(a)}(n), \mathrm{E}_{\mathrm{E}_k(a)}(n))$*. Since we cannot derive* $(k,k) \in \mathcal{SA}(h' \cup \mathrm{Id}_B)$ *we must have* $(\mathrm{E}_k(a), \mathrm{E}_k(a)) \in \mathcal{A}(h' \cup \mathrm{Id}_B)$*. This is the case iff* $M = a = N$*, yielding* $\sigma' = \{^{a}/_{x}\} = \rho'$*.*

Finally, se'' *is concretizable since* $\mathrm{dom}(th') \subset \mathcal{M} \times \mathcal{M}$ *and consistent since it is symmetric.*

4.3.2 A Simple Cryptographic Protocol

We consider the simple cryptographic protocol

$$(\nu k)\,(A \mid B) \quad \text{where} \quad A := \overline{a}\langle \mathrm{E}_k(m)\rangle \quad \text{and} \quad B := a(x).\overline{f}\langle \mathrm{D}_k(x)\rangle$$

consisting of the participant A sending on channel a the message m, encrypted under the secret shared symmetric key k, to the participant B who tries to decrypt the received message and, in case of successful decryption, outputs the result on channel f. We may compare this protocol with its *specification*

$$(\nu k)\,(\underline{A} \mid \underline{B}) \quad \text{where} \quad \underline{A} := \overline{a}\langle \mathrm{E}_k(m)\rangle \quad \text{and} \quad \underline{B} := a(y).[\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle$$

where $\underline{B}$ transmits the correct message m on channel f whenever the dummy message (on reception bound to y) can be decrypted (as expressed by the guard $[\mathrm{D}_k(y) : \mathcal{M}]$). If the equation $(\nu k)\,(\underline{A} \mid \underline{B}) = (\nu k)\,(A \mid B)$ holds, then no context is able to influence the authenticity (more precisely: integrity) of the message m.

We start with a symbolic environment in which the message m is a variable: We let $th := \{(a,a) \mapsto 0, (f,f) \mapsto 0\}$, $tv := \{(m,m) \mapsto 1\}$ and $se := (th, tv, (tt, tt))$. Note that we give m a later time than a and f, in order to permit occurrences of a and f in the message.

Proposition 4.3.4 $se \vdash (\nu k)\,(\underline{A} \mid \underline{B}) \sim_{\mathrm{s}} (\nu k)\,(A \mid B)$

Proof: We let $g_{F(x)} := [\,F : \mathcal{N}\,]$, $g^{\overline{F}G} := [\,F : \mathcal{N}\,] \wedge [\,G : \mathcal{M}\,]$ and $g^{\overline{F'}G}_{F(x)} := g_{F(x)} \wedge g^{\overline{F'}G} \wedge [\,F = F'\,]$. We write $\mathrm{pwd}(\tilde{u})$ to denote that $\tilde{u}$ is a tuple of *p*air-*w*ise *d*ifferent names and/or variables.

The specification process $(\nu k)\,(\underline{A} \mid \underline{B})$ and its derivatives have their symbolic transitions given in Table 4.2. The process $(\nu k)\,(A \mid B)$ and its derivatives have their symbolic transitions given in Table 4.3. The symmetric closure of the set $\mathcal{R}$ defined in Table 4.4 is a symbolic bisimulation.

Note that $\mathcal{R}$ is infinite, but that this infinity only arises from the possible different choices of bound names. Effectively, the bisimulation contains only $8 \cdot 2 = 16$ process pairs. We only check the fourth element in $\mathcal{R}$, namely

$$\begin{aligned}((th, tv \cup \{x \mapsto 2, y_1 \mapsto 2\}, ([\,y_1 = a\,], [\,y_1 = a\,])),& \\ (\nu k)\,(\underline{A} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle),\ (\nu k)\,(A \mid \overline{f}\langle \mathrm{D}_k(x)\rangle))&\end{aligned}$$

of which we denote the environment by se'.

Consistency se' is clearly well-timed. se' is concretizable since $\mathrm{dom}(th)$ only contains messages. $\mathbf{C}^B_{\sigma,\rho}(th) = \mathrm{Id}_B \cup \{(a,a),(f,f)\}$ whenever defined, which is consistent since $\{a,f\} \cap B = \emptyset$.

$$(\nu k)\,(\underline{A} \mid \underline{B}) \xrightarrow[(\nu kn)\, g_{a(y)}^{\overline{a}\ \mathrm{E}_k(n)}]{\tau} (\nu kn)\,(\mathbf{0} \mid [\mathrm{D}_k(\mathrm{E}_k(n)) : \mathcal{M}]\overline{f}\langle m\rangle)$$

$$(\nu k)\,(\underline{A} \mid \underline{B}) \xrightarrow[g^{\overline{a}\ \mathrm{E}_k(n)}]{(\nu kn)\,\overline{a}\ \mathrm{E}_k(n)} \mathbf{0} \mid \underline{B}$$

$$(\nu k)\,(\underline{A} \mid \underline{B}) \xrightarrow[g_{a(y)}]{a(y)} (\nu k)\,(\underline{A} \mid \overline{f}\langle \mathrm{D}_k(y)\rangle)$$

$$(\nu kn)\,(\mathbf{0} \mid [\mathrm{D}_k(\mathrm{E}_k(n)) : \mathcal{M}]\overline{f}\langle m\rangle) \xrightarrow[(\nu k'n')\, g^{\overline{f}\ m \wedge [\mathrm{D}_{k'}(\mathrm{E}_{k'}(n')) : \mathcal{M}]}]{\overline{f}\ m} (\nu kn)\,(\mathbf{0} \mid \mathbf{0})$$

$$\mathbf{0} \mid \underline{B} \xrightarrow[g_{a(y)}]{a(y)} \mathbf{0} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle$$

$$\mathbf{0} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle \xrightarrow[g^{\overline{f}\ m \wedge [\mathrm{D}_k(y) : \mathcal{M}]}]{\overline{f}\ m} \mathbf{0} \mid \mathbf{0}$$

$$(\nu k)\,(\underline{A} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle) \xrightarrow[g^{\overline{a}\ \mathrm{E}_k(n)}]{(\nu kn)\,\overline{a}\ \mathrm{E}_k(n)} \mathbf{0} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle$$

$$(\nu k)\,(\underline{A} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle) \xrightarrow[(\nu k'n')\, g^{\overline{f}\ m \wedge [\mathrm{D}_{k'}(\mathrm{E}_{k'}(n')) : \mathcal{M}]}]{\overline{f}\ m} (\nu k)\,(\underline{A} \mid \mathbf{0})$$

$$(\nu k)\,(\underline{A} \mid \mathbf{0}) \xrightarrow[g^{\overline{a}\ \mathrm{E}_k(n)}]{(\nu kn)\,\overline{a}\ \mathrm{E}_k(n)} \mathbf{0} \mid \mathbf{0}$$

Table 4.2: Symbolic Transitions of Specification Process

$$(\nu k)\,(A \mid B) \xrightarrow[(\nu k)\, g_{a(y)}^{\overline{a}\ \mathrm{E}_k(m)}]{\tau}_s (\nu k)\,(\mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(\mathrm{E}_k(m))\rangle)$$

$$(\nu k)\,(A \mid B) \xrightarrow[g^{\overline{a}\ \mathrm{E}_k(m)}]{(\nu k)\,\overline{a}\ \mathrm{E}_k(m)}_s \mathbf{0} \mid B$$

$$(\nu k)\,(A \mid B) \xrightarrow[g_{a(y)}]{a(y)}_s (\nu k)\,(A \mid \overline{f}\langle \mathrm{D}_k(y)\rangle)$$

$$(\nu k)\,(\mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(\mathrm{E}_k(m))\rangle) \xrightarrow[(\nu k')\, g^{\overline{f}\ \mathrm{D}_{k'}(\mathrm{E}_{k'}(m))}]{\overline{f}\, m}_s (\nu k)\,(\mathbf{0} \mid \mathbf{0})$$

$$\mathbf{0} \mid B \xrightarrow[g_{a(y)}]{a(y)}_s \mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(y)\rangle$$

$$\mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(y)\rangle \xrightarrow[g^{\overline{f}\ \mathrm{D}_k(y)}]{\overline{f}\ \mathrm{D}_k(y)}_s \mathbf{0} \mid \mathbf{0}$$

$$(\nu k)\,(A \mid \overline{f}\langle \mathrm{D}_k(y)\rangle) \xrightarrow[g^{\overline{a}\ \mathrm{E}_k(m)}]{(\nu k)\,\overline{a}\ \mathrm{E}_k(m)}_s \mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(y)\rangle$$

$$(\nu k)\,(A \mid \overline{f}\langle \mathrm{D}_k(y)\rangle) \xrightarrow[(\nu k')\, g^{\overline{f}\ \mathrm{D}_{k'}(y)}]{\overline{f}\, m}_s (\nu k)\,(A \mid \mathbf{0})$$

$$(\nu k)\,(A \mid \mathbf{0}) \xrightarrow[g^{\overline{a}\ \mathrm{E}_k(m)}]{(\nu k)\,\overline{a}\ \mathrm{E}_k(m)}_s \mathbf{0} \mid \mathbf{0}$$

Table 4.3: Symbolic Transitions of Implementation Process

$$
\begin{array}{l}
\mathcal{R} := \{((th, tv, tt, tt), (\nu k)\,(\underline{A} \mid \underline{B}),\ (\nu k)\,(A \mid B)),\\
\\
\quad ((th, tv, ((\nu k)\ tt, (\nu k)\ tt)),\\
\quad (\nu k)\,(\mathbf{0} \mid [\mathrm{D}_k(\mathrm{E}_k(m)) : \mathcal{M}]\overline{f}\langle m\rangle),\ (\nu k)\,(\mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(\mathrm{E}_k(m))\rangle)),\\
\\
\quad ((th \cup \{(m, \mathrm{D}_k(\mathrm{E}_k(m))) \mapsto 2\}, tv \cup \{y_1 \mapsto 2\},\\
\quad\ ((\nu\{k, k'\})\,[\,y_1 = f\,], (\nu\{k, k'\})\,[\,y_1 = f\,])),\\
\quad (\nu k)\,(\mathbf{0} \mid \mathbf{0}),\ (\nu k)\,(\mathbf{0} \mid \mathbf{0})),\\
\\
\quad ((th, tv \cup \{x \mapsto 2, y_1 \mapsto 2\}, ([\,y_1 = a\,], [\,y_1 = a\,])),\\
\quad (\nu k)\,(\underline{A} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle),\ (\nu k)\,(A \mid \overline{f}\langle \mathrm{D}_k(x)\rangle)),\\
\\
\quad ((th \cup \{(\mathrm{E}_k(m), \mathrm{E}_k(m)) \mapsto 3\}, tv \cup \{x \mapsto 2, y_1 \mapsto 2, y_2 \mapsto 3\},\\
\quad\ ((\nu k)\,[\,y_1 = a\,] \wedge [\,y_2 = a\,], (\nu k)\,[\,y_1 = a\,] \wedge [\,y_2 = a\,])),\\
\quad (\mathbf{0} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle),\ (\mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(x)\rangle)),\\
\\
\quad ((th \cup \{(\mathrm{E}_k(m), \mathrm{E}_k(m)) \mapsto 2\}, tv \cup \{y_1 \mapsto 2\},\\
\quad\ ((\nu k)\,[\,y_1 = a\,], (\nu k)\,[\,y_1 = a\,])),\ (\mathbf{0} \mid \underline{B}),\ (\mathbf{0} \mid B)),\\
\\
\quad ((th \cup \{(\mathrm{E}_k(m), \mathrm{E}_k(m)) \mapsto 2\}, tv \cup \{y_1 \mapsto 2, y_2 \mapsto 3, x \mapsto 3\},\\
\quad\ ((\nu k)\,[\,y_1 = a\,] \wedge [\,y_2 = a\,], (\nu k)\,[\,y_1 = a\,] \wedge [\,y_2 = a\,])),\\
\quad (\mathbf{0} \mid [\mathrm{D}_k(y) : \mathcal{M}]\overline{f}\langle m\rangle),\ (\mathbf{0} \mid \overline{f}\langle \mathrm{D}_k(x)\rangle)),\\
\\
\quad ((th \cup \{(\mathrm{E}_k(m), \mathrm{E}_k(m)) \mapsto 2, (m, \mathrm{D}_k(x)) \mapsto 4\},\\
\quad\ tv \cup \{y_1 \mapsto 2, y_2 \mapsto 3, x \mapsto 3, y_3 \mapsto 4\},\\
\quad\ ((\nu k)\,[\,y_1 = a\,] \wedge [\,y_2 = a\,] \wedge [\,y_3 = f\,] \wedge [\mathrm{D}_k(x) : \mathcal{M}],\\
\quad\ \ (\nu k)\,[\,y_1 = a\,] \wedge [\,y_2 = a\,] \wedge [\,y_3 = f\,] \wedge [\mathrm{D}_k(x) : \mathcal{M}]),\\
\quad (\mathbf{0} \mid \mathbf{0}),\ (\mathbf{0} \mid \mathbf{0}))\\
\quad \mid \mathrm{pwd}(a, f, m, x, y_1, y_2, y_3, k, k')\}
\end{array}
$$

Table 4.4: A Symbolic Bisimulation

Assume that $(th, tv \cup \{x \mapsto 2, y_1 \mapsto 2\}, (tt, tt)) \vdash \sigma \leftrightarrow_B \rho$. By the symmetry of se' we need only consider the case $[\![[\, y_1 = a \,]\sigma]\!]$. Then $\sigma(y_1) = a$, and by the consistency of $\mathbf{C}^B_{\sigma,\rho}(th)$ we get $\rho(y_1) = a$.
Thus se' is consistent.

Transition 1 $(\nu k)\,(\underline{A} \,|\, [\,\mathrm{D}_k(y) : \mathcal{M}\,]\overline{f}\langle m\rangle) \xrightarrow[g^{\overline{a}\,\mathrm{E}_k(m)}]{(\nu k)\,\overline{a}\,\mathrm{E}_k(m)}_s \mathbf{0} \,|\, [\,\mathrm{D}_k(y) : \mathcal{M}\,]\overline{f}\langle m\rangle$ has to be simulated, since if we let $\sigma' := \rho' := \left\{{}^{a}/_{m}\right\}\left\{{}^{a}/_{x}\right\}\left\{{}^{a}/_{y_1}\right\}$ then
$se' \vdash \sigma' \leftrightarrow_\emptyset \rho'$ and $a \in \{a, f\} = \pi_1(\mathbf{C}^\emptyset_{\sigma',\rho'}(th))$. Now

$$\{(th, tv \cup \{x \mapsto 2, y_1 \mapsto 2, y_2 \mapsto 3\}, \\ ((\nu k)\,[\, y_1 = a \,] \wedge [\, y_2 = a \,], [\, y_1 = a \,] \wedge [\, y_2 = a \,]))\}$$

is a decomposition of

$$(th, tv \cup \{x \mapsto 2, y_1 \mapsto 2, y_2 \mapsto 3\}, \\ ((\nu k)\,[\, y_1 = a \,] \wedge [\, y_2 = a \,], [\, y_1 = a \,])).$$

We simulate the transition by $(\nu k)\,(A \,|\, \overline{f}\langle \mathrm{D}_k(x)\rangle) \xrightarrow[g^{\overline{a}\,\mathrm{E}_k(m)}]{(\nu k)\,\overline{a}\,\mathrm{E}_k(m)}_s \mathbf{0} \,|\, \overline{f}\langle \mathrm{D}_k(x)\rangle$.
We let

$$se'' = (th, tv \cup \{x \mapsto 2, y_1 \mapsto 2, y_2 \mapsto 3\}, \\ ((\nu k)\,[\, y_1 = a \,] \wedge [\, y_2 = a \,], (\nu k)\,[\, y_1 = a \,] \wedge [\, y_2 = a \,]))$$

Then $se'' \vdash tt \leftrightarrow g^{\overline{a}\,\mathrm{E}_k(m)} \wedge [\, y_2 = a \,]$ since $g^{\overline{a}\,\mathrm{E}_k(m)}\sigma = [\, a : \mathcal{N} \,] \wedge \mathcal{M}\mathrm{E}_k(\sigma(m))$ holds.

Transition 2 First we α-rename to avoid clashes with environment names.
$(\nu k')\,(\underline{A} \,|\, [\,\mathrm{D}_{k'}(y) : \mathcal{M}\,]\overline{f}\langle m\rangle) \xrightarrow[g^{\overline{f}\,m \wedge [\mathrm{D}_{k'}(y) : \mathcal{M}]}]{(\nu k')\,\overline{f}\,m}_s (\nu k)\,(\underline{A} \,|\, \mathbf{0})$ does not need to be simulated: $[\![[\,\mathrm{D}_{k'}(\sigma(y)) : \mathcal{M}\,]]\!]$ holds iff $\sigma(y) = \mathrm{E}_{k'}(M)$ for some M, but k' cannot be in $\mathrm{n}(\mathrm{range}(\sigma))$ since it is bound in the transition.

□

Chapter 5

Conclusions

We set out to address the problem of automatic verification of observational equivalence in channel-passing calculi with expression languages.

In Chapter 2, we studied the different environment-sensitive bisimilarities that had been defined for the spi calculus. We developed a general framework for comparing environment-sensitive relations, and showed how to leverage counterexamples to yield impossibility results in this framework. We gained a good understanding of how to define environment-sensitive relations without introducing unnecessary incompleteness.

Our hedged bisimulation has been used in more recent work [Tiu07, Bri08, KM07]. For future work in this area, the question of whether to work in a "hedged" or an "alley" style remains open, but the latter seems more straightforward when working with complex expression languages.

In Chapter 3, we investigated extensions of the spi calculus expression language. We gave the first full proof that static equivalence is harder to decide than knowledge, in a simpler setting than [AC06]. We defined a class of constructor-destructor expression languages that enabled a smooth extension of proof techniques for the simple spi expression language of Section 1.4. We characterized synthesis and consistency over these constructor-destructor languages in terms of knowledge and static equivalence over a related language. Besides validating our definitions, this investigation also makes more precise the equivalence hinted at in [AF01]. Finally, we made a significant generalization of the concept of spi calculus, permitting any expression language with a deterministic evaluation function satisfying some natural constraints.

Finally, in Chapter 4 we addressed the problem of infinite branching on process input. We defined a general symbolic operational semantics for any spi calculus. Restricting ourselves to the constructor-destructor languages, we defined symbolic environments and decompositions of them, and showed through an example that infinite distinctions or powerful logics are needed even for binary case splits. We

defined a symbolic bisimilarity and proved it sound and complete with respect to concrete hedged bisimilarity, and thus also with respect to barbed equivalence.

Baudet [Bau07] proved that symbolic consistency is decidable for positive guards and subterm-convergent languages; One possibility for future work is to investigate whether these techniques can yield decidability also for guards with negation, at least in the simpler case of constructor-destructor languages.

Another possibility for future work is to redefine the notion of concretization of a symbolic environment to instead yield a pair of statically equivalent substitutions, allowing symbolic bisimilarity to be applied to all expression languages with a notion of evaluation rather than just the constructor-destructor languages.

Finally, to reach the goal of fully automated verification one could investigate heuristics, logics and/or language restrictions for generating finite decompositions of symbolic environments.

Acknowledgments

Thanks first to Uwe, for providing me with the opportunity to do a PhD in the first place, as well as many other opportunities along the way. Thanks to my office mates at EPFL: Rachele, for many alpine hikes, Cleo, for her enthusiasm, Sébastien, for challenging my assumptions, definitions and proofs, and Simon, for the example he set. The lampions and the basement republic provided a very social climate. Martín Abadi graciously hosted my work on Section 3.2. Andy Gordon supervised me during a summer internship at Microsoft Research in Cambridge, which was a great experience. Many thanks to Sonja, last but not least for supporting me while I was writing up.

During my PhD, I received funding from the Swiss NSF project "Theory and Tool Support for the Formal Verification of Cryptographic Protocols", grants no. 200020-101720.1 and 21-65180.1, and the EU project PEPITO (IST-2001-33234).

Appendix A

Proofs

A.1 Proofs of Chapter 3

We first extend the notion of synthesis in order to be able to generate non-message expressions from a hedge, and prove some results for consistent hedges on this extended synthesis.

Definition A.1.1 *If h is a hedge, we let $\mathcal{S}'(h)$ be the smallest set containing h and satisfying the following rule.*

$$(\text{SYN}') \ \frac{(F_j, G_j) \in \mathcal{S}'(h) \text{ for } j \in \{1, \ldots, \mathrm{ar}(f)\}}{(f(\widetilde{F}), f(\widetilde{G})) \in \mathcal{S}'(h)} \quad \begin{array}{l} f \in \mathcal{F}' \\ f(\widetilde{F}) \not\to^{\mathrm{H}}_{E'} \\ f(\widetilde{G}) \not\to^{\mathrm{H}}_{E'} \end{array}$$

Lemma A.1.2 *If h is left consistent then*

1. *If $(F, G), (F', G') \in \mathcal{S}'(h)$ and $F = F'$ then $G = G'$.*
2. *If $f \in \mathcal{F}'$ and $(F^1_j, F^2_j) \in \mathcal{S}'(h)$ for $j \in \{1, \ldots, \mathrm{ar}(f)\}$ with $f(\widetilde{F}^1) \to^{\mathrm{H}}_{E'} F^1$ then $f(\widetilde{F}^2) \to^{\mathrm{H}}_{E'} F^2$ and $(F^1, F^2) \in \mathcal{S}'(h)$.*

Proof:

1. By induction on the shortest derivation of either of $(F, G) \in \mathcal{S}'(h)$ and $(F', G') \in \mathcal{S}'(h)$. By symmetry we may assume that this is a derivation of $(F, G) \in \mathcal{S}'(h)$.
   If $(F, G) = (M, N) \in h$ and $F' = M$ then there are two possibilities. If $(M, G') \in h$ then $G = G'$ by condition 2 of Definition 3.3.20. Otherwise, since M only contains function symbols in $\mathcal{F}^+$, we have $(M, G') \in \mathcal{S}^+(h)$ which yields a contradiction to condition 3.
   If we used SYN$'$ to derive $(F, G) \in \mathcal{S}'(h)$ we also used SYN$'$ to derive $(F', G') \in \mathcal{S}'(h)$, so there are $f \in \mathcal{F}'$ and $(F_j, G_j), (F'_j, G'_j) \in \mathcal{S}'(h)$ for

$j \in \{1, \ldots, \mathrm{ar}(f)\}$ such that $(F, G) = (f(\widetilde{F}), f(\widetilde{G}))$ and $(F', G') = (f(\widetilde{F'}), f(\widetilde{G'}))$. Since $F = F'$ we have $F_j = F'_j$ for every j, so by induction every $G_j = G'_j$ and thus $G = G'$.

2. We begin by a case analysis on $f \in \mathcal{F}'$.

   $f \in \mathcal{F}^+ \cup \{\mathtt{OK}\}$: In this case, $f(\widetilde{F}) \not\to^{\mathrm{H}}$.

   $f = \mathtt{name}$: If $(F, G) \in \mathcal{S}'(h)$ and $\mathtt{name}(F) \to^{\mathrm{H}} F$, then $F \in \mathcal{N}$, so SYN$'$ cannot have been used to derive $(F, G) \in \mathcal{S}'(h)$. Thus $(F, G) \in h$, so $G \in \mathcal{N}$ by condition 1 for consistency. Then $\mathtt{name}(G) \to^{\mathrm{H}} G$, and by assumption $(F, G) \in \mathcal{S}'(h)$.

   $f = f_{ijk}^{-1} \in \mathcal{F}^-$: Let σ_1, σ_2 be injective with $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2)$ and $h = \{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\}$. Then for each l there is F_l with $F_l^1 = F_l\sigma_1 \not\to_E$ and $F_l^2 = F_l\sigma_2 \not\to_E$.
   We first show that if $f_{ijk}^{-1}(\widetilde{F^1}) \to^{\mathrm{H}} F^2$ for some F^2, then $(F^1, F^2) \in \mathcal{S}'(h)$. If $j > \mathrm{ar}(f_i)$ we have $(F^1, F^2) = (F^1_{j+1-\mathrm{ar}(f_i)}, F^2_{j+1-\mathrm{ar}(f_i)}) \in \mathcal{S}'(h)$. Otherwise, if $F_1^1 \in \pi_1(h)$ then $(F_1^1, F_1^2) \in h$ by condition 3 for consistency. Since LHS_{ijk} only contains function symbols in $\mathcal{F}^+$ there are $(M_l, N_l) \in \mathcal{S}(h)$ such that $f_{ijk}^{-1}(F_1^1, \widetilde{M}) \to^{\mathrm{H}} F^1$ and $f_{ijk}^{-1}(F_1^2, \widetilde{N}) \to^{\mathrm{H}} F^2$. Then $(F^1, F^2) \in \mathcal{A}(h)$ by ANA, so by Lemma 3.3.12.4 and Lemma 3.3.10 we have $(F^1, F^2) \in \mathcal{S}(h) \subseteq \mathcal{S}'(h)$. Otherwise, SYN$'$ was used to derive $(F_1^1, F_1^2) \in \mathcal{S}'(h)$, so $(F^1, F^2) \in \mathcal{S}'(h)$.
   To show that $f_{ijk}^{-1}(\widetilde{F^1}) \to^{\mathrm{H}}$, we adapt a proof of [AC04a] (Lemma 2) to the present setting. We let $G = f_{ijk}^{-1}(\widetilde{N})$ be a greatest expression with $\mathrm{v}(G, \mathrm{LHS}_{ijk})$ fresh (i.e., not present in any expression previously mentioned) and $\mathrm{v}(G) \cap \mathrm{v}(\mathrm{LHS}_{ijk}) = \emptyset$ such that there are substitutions $\rho', \rho'' : \mathrm{v}(G) \to \mathcal{E}$ with $G\rho' = \mathrm{LHS}_{ijk}$ and $G\rho'' = F$. By 1, for all $x, y \in \mathrm{v}(G)$ with $x\rho''\sigma_1 = y\rho''\sigma_1$ we have $x\rho''\sigma_2 = y\rho''\sigma_2$. We divide $\mathrm{v}(G)$ into three disjoint sets as follows:

$$\begin{aligned}\mathrm{v}(G) = {} & \{x_i \mid \rho'(x_i) \notin \mathcal{V}\} \\ & \uplus \{y_i \mid \rho'(y_i) \in \mathcal{V} \wedge \rho'(y_i) \notin \mathrm{v}(\mathrm{range}(\rho') \setminus \mathcal{V})\} \\ & \uplus \{z_i \mid \rho'(z_i) \in \mathcal{V} \wedge \exists x_{z_i} \in \{x_i\} \mid \rho'(z_i) \in \mathrm{v}(\rho'(x_{z_i}))\}\end{aligned}$$

   Assume that $\rho_1 : \mathrm{v}(\mathrm{LHS}_{ijk}) \to \mathcal{E}$ satisfies $f_{ijk}^{-1}(\widetilde{F^1}) = \mathrm{LHS}_{ijk}\,\rho_1$. For any x_i, there is $t_{x_i} \in \mathrm{dom}(\sigma_1)$ with $\rho''(x_i) = t_{x_i}$. For any z_i, we have that $\rho_1(\rho'(z_i))$ is a subterm of $\sigma_1(\rho''(x_{z_i}))$. We let $\gamma = \{x_i \mapsto \rho''(x_i)\} \cup \{y_i \mapsto \rho'(y_i)\} \cup \{z_i \mapsto \rho''(z_i)\}$ and consider $G\gamma$.
   First, if $\rho'_1 = \{x \mapsto \rho_1(x) \mid x \in \mathrm{dom}(\rho_1) \setminus \{\rho'(z_i)\}\}$ then $G\gamma\sigma_1 = \mathrm{LHS}_{ijk}\,\rho'_1$, so $G\gamma\sigma_1 \to^{\mathrm{H}}_E$. Then, $G\gamma$ is a $\pi_1(h)$-pattern, so $G\gamma\sigma_2 \to^{\mathrm{H}}_E$ and thus $G\rho''\sigma_2 \to^{\mathrm{H}}_E$.

$f = f^?_{ijk} \in \mathcal{F}^?$: Assume that $f^?_{ijk}(\widetilde{F}^1) \to^{\mathrm{H}} F^1$, i.e., $F^1_{\mathrm{ar}(f^?_{ijk})} = F^1 = \mathtt{OK}$ and $f^{-1}_{ijk}F^1_1, \ldots, F^1_{\mathrm{ar}(f^?_{ijk})-1} \to^{\mathrm{H}}$. As above $f^{-1}_{ijk}F^2_1, \ldots, F^2_{\mathrm{ar}(f^?_{ijk})-1} \to^{\mathrm{H}}$. Since $F^1_{\mathrm{ar}(f^?_{ijk})} = \mathtt{OK} \notin \mathcal{M}$ then $F^2_{\mathrm{ar}(f^?_{ijk})} = \mathtt{OK}$, so $f^?_{ijk}\widetilde{F}^2 \to^{\mathrm{H}} \mathtt{OK}$. By SYN′ we have $(\mathtt{OK}, \mathtt{OK}) \in \mathcal{S}'(h)$.

□

We can then extend the result of Theorem 3.3.17 to the present setting.

Lemma A.1.3 *If* $\sigma_1, \sigma_2 : \mathcal{V} \rightharpoonup \mathcal{M}$ *with* $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2)$, *and* $h = \{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\}$ *then*

1. *if* $(M_1, M_2) \in \mathcal{S}(\mathcal{I}(h))$ *then*
   there is $F \in \mathcal{T}_\Sigma$ *with* $\mathrm{n}(F) = \emptyset$, $\mathbf{e}(F\sigma_1) = M_1$ *and* $\mathbf{e}(F\sigma_2) = M_2$*; and*
2. *if* $(F_1, F_2) \in \mathcal{S}'(\mathcal{I}(h))$ *then*
   there is $F \in \mathcal{T}_\Sigma$ *with* $\mathrm{n}(F) = \emptyset$, $F\sigma_1{\downarrow} = F_1$ *and* $F\sigma_2{\downarrow} = F_2$*; and*
3. *if* h *is consistent and* $F \in \mathcal{T}_{\Sigma'}$ *with* $\mathrm{n}(F) \cap \mathrm{n}(h) = \emptyset$ *then*
   $(F\sigma_1{\downarrow}, F\sigma_2{\downarrow}) \in \mathcal{S}'(h \cup \mathrm{Id}_{\mathrm{n}(F)})$.

Proof.

1. By Lemma 3.3.10 we have that $\mathcal{A}(h) \subseteq \mathcal{S}(\mathcal{I}(h)) \subseteq \mathcal{S}'(\mathcal{I}(h))$. We prove that whenever $(M_1, M_2) \in \mathcal{S}'(\mathcal{A}(h))$, there is $F \in \mathcal{T}_\Sigma$ with $F\sigma_1{\downarrow}_E = M_1$ and $F\sigma_2{\downarrow}_E = M_2$ by induction on the derivation. If $(M_1, M_2) \in h$ there is $x \in \mathrm{dom}(\sigma_1)$ such that $M_1 = \sigma_1(x)$ and $M_2 = \sigma_2(x)$.
   If $(M_1, M_2) \in \mathcal{A}(h)$ was derived using ANA, there are $f^{-1}_{ijk}, \widetilde{N}^1, \widetilde{N}^2$ with $f^{-1}_{ijk}(\widetilde{N}^1) \to^{\mathrm{H}}_E M_1$, $f^{-1}_{ijk}(\widetilde{N}^2) \to^{\mathrm{H}}_E M_2$ and $(N^1_j, N^2_j) \in \mathcal{S}(\mathcal{A}(h))$ for $j \in \{1, \ldots, \mathrm{ar}(f^{-1}_{ijk})\}$. By induction we get $\widetilde{G}$ such that $\widetilde{N}^1 = \widetilde{\mathbf{e}(G\sigma_1)}$ and $\widetilde{N}^2 = \widetilde{\mathbf{e}(G\sigma_2)}$. Then $M_1 = \mathbf{e}(f^{-1}_{ijk}(\widetilde{G})\sigma_1)$ and $M_2 = \mathbf{e}(f^{-1}_{ijk}(\widetilde{G})\sigma_2)$.
   If $(M_1, M_2) \in \mathcal{S}'(\mathcal{A}(h))$ was derived using SYN, there are $f_i, \widetilde{N}^1, \widetilde{N}^2$ with $M_1 = f_i(\widetilde{N}^1)$, $M_2 = f_i(\widetilde{N}^2)$ and $(N^1_j, N^2_j) \in \mathcal{S}(\mathcal{A}(h))$ for $j \in \{1, \ldots, \mathrm{ar}(f_i)\}$. By induction we get $\widetilde{G}$ such that $\widetilde{N}^1 = \widetilde{\mathbf{e}(G\sigma_1)}$ and $\widetilde{G}^2 = \widetilde{\mathbf{e}(G\sigma_2)}$. Then $M_1 = \mathbf{e}(f_i(\widetilde{G})\sigma_1)$ and $M_2 = \mathbf{e}(f_i(\widetilde{G})\sigma_2)$.
2. We prove the statement by induction on the derivation of $(F_1, F_2) \in \mathcal{S}'(\mathcal{I}(h))$. By 1, whenever $(F_1, F_2) \in \mathcal{I}(h)$ there is $F \in \mathcal{T}_\Sigma$ with $\mathrm{n}(F) = \emptyset$, $F_1 = \mathbf{e}(F\sigma_1) = F\sigma_1{\downarrow}$ and $F_2 = \mathbf{e}(F\sigma_2) = F\sigma_2{\downarrow}$.
   If $(F_1, F_2) \in \mathcal{S}'(\mathcal{A}(h))$ was derived using SYN′, there are $f, \widetilde{G}^1, \widetilde{G}^2$ with $F_1 = f(\widetilde{G}^1) \not\to^{\mathrm{H}}_{E'}$, $F_2 = f(\widetilde{G}^2) \not\to^{\mathrm{H}}_{E'}$ and $(G^1_j, G^2_j) \in \mathcal{S}'(\mathcal{A}(h))$ for $j \in \{1, \ldots, \mathrm{ar}(f_i)\}$. By induction we get $\widetilde{G}$ such that $\widetilde{G}^1 = \widetilde{G}\sigma_1{\downarrow}$ and $\widetilde{G}^2 = \widetilde{G}\sigma_2{\downarrow}$. Then $F_1 = f_i(\widetilde{G}\sigma_1{\downarrow}) = f_i(\widetilde{G})\sigma_1{\downarrow}$ and $F_2 = f_i(\widetilde{G}\sigma_2{\downarrow}) = f_i(\widetilde{G})\sigma_2{\downarrow}$.

3. By Lemma 3.3.22 $h \cup \mathrm{Id}_{\mathrm{n}(F)} = g$ is consistent. We prove the statement by induction on F. If $F = a$ is a name, then $F\sigma_1 = a = F\sigma_2$ and $(a, a) \in \mathrm{Id}_{\mathrm{n}(a)} \subseteq \mathcal{S}'(g)$. If $F = x \in \mathrm{dom}(\sigma_1)$ is a variable, then $(F\sigma_1, F\sigma_2) \in h \subseteq \mathcal{S}'(g)$.
Otherwise, there are are $f \in \mathcal{F}'$ and G_j for $j \in \{1, \ldots, \mathrm{ar}(f)\}$ such that $F = f(\widetilde{G})$. By induction $(G_i\sigma_1{\downarrow}, G_i\sigma_2{\downarrow}) \in \mathcal{S}'(g)$. The statement then follows directly from Lemma A.1.2.2 applied to g and g^{T}.

□

We can now prove Theorem 3.3.24, restated below.

Theorem A.1.4 (Theorem 3.3.24) *If $\sigma_1, \sigma_2 : \mathcal{V} \rightharpoonup \mathcal{M}$ with $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\sigma_2)$, then $\sigma_1 \cong_{E'} \sigma_2$ iff $\mathcal{I}(\{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\})$ is consistent.*

Proof:

$\Rightarrow$: If $\sigma_1 \cong_{E'} \sigma_2$ then $h = \mathcal{I}(\{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\})$ is irreducible by Lemma 3.3.12.4. We test the four conditions of semi-consistency.

1. Assume that $(a, N) \in h$ with $a \in \mathcal{N}$. By Lemma A.1.3 there is F with $\mathrm{n}(F) = \emptyset$, $F\sigma_1{\downarrow} = a$ and $F\sigma_2{\downarrow} = N$. Let $G = \mathtt{name}(F)$. We have $F\sigma_1{\downarrow} = a = G\sigma_1{\downarrow}$, so $N = F\sigma_2{\downarrow} = G\sigma_2{\downarrow} = \mathtt{name}(N){\downarrow}$, which holds iff $N \in \mathcal{N}$.

2,3 Assume that $(M, N) \in h$ and $(M', N') \in \mathcal{S}(h)$ such that $M = M'$. By Lemma A.1.3 there are F, G with $\mathrm{n}(F, G) = \emptyset$, $(F\sigma_1{\downarrow}, F\sigma_2{\downarrow}) = (M, N)$ and $(G\sigma_1{\downarrow}, G\sigma_2{\downarrow}) = (M', N')$. By static equivalence $N = N'$. By monotonicity $\mathcal{S}^+(\mathcal{I}(h)) \subseteq \mathcal{S}^+(\mathcal{A}(h))$, and since $\mathcal{I}(h) \cap \mathcal{S}^+(\mathcal{A}(h)) = \emptyset$ we cannot have $(M, N) \in \mathcal{S}^+(\mathcal{I}(h))$.

4 Take ρ_1, ρ_2 with $\mathcal{I}(h) = \{(\rho_1(x), \rho_2(x)) \mid x \in \mathrm{dom}(\rho_1)\}$ and $\mathrm{dom}(\rho_1) = \mathrm{dom}(\rho_1)$. Let $f_{ijk}^{-1}(\widetilde{M})$ be a ρ_1-pattern and assume that $f_{ijk}^{-1}(\widetilde{M})\rho_1 \rightarrow^{\mathrm{H}} M'$.
By Lemma A.1.3 there is for each $x \in \mathrm{dom}(\rho_1)$ an expression F_x with $\mathrm{n}(F_x) = \emptyset$, $F_x\sigma_1{\downarrow} = \rho_1(x)$ and $F_x\sigma_2{\downarrow} = \rho_2(x)$. We let $\gamma = \{x \mapsto F_x \mid x \in \mathrm{dom}(\rho_1)\}$ and $F = f_{ijk}^{-1}(\widetilde{M})\gamma$.
By Lemma A.1.3 we have $(F\sigma_1{\downarrow}, F\sigma_2{\downarrow}) \in \mathcal{S}'(h)$. Since $F\sigma_1{\downarrow} = M' \in \mathcal{M}$ we must have $(F\sigma_1{\downarrow}, F\sigma_2{\downarrow}) \in \mathcal{S}(h)$, so $F\sigma_2{\downarrow} \in \mathcal{M}$ and thus $f_{ijk}^{-1}(\widetilde{M})\rho_2 \rightarrow^{\mathrm{H}}$.

The consistency of h follows by symmetry.

$\Leftarrow$: Assume that $h = \mathcal{I}(\{(\sigma_1(x), \sigma_2(x)) \mid x \in \mathrm{dom}(\sigma_1)\})$ is consistent. Take any F, G with $\mathrm{n}(F, G) \cap \mathrm{n}(\sigma_1, \sigma_2) = \emptyset$. Then $g = h \cup \mathrm{Id}_{\mathrm{n}(F,G)}$ is consistent by Lemma 3.3.22. We also have $(F\sigma_1{\downarrow}, F\sigma_2{\downarrow}), (G\sigma_1{\downarrow}, G\sigma_2{\downarrow}) \in \mathcal{S}'(g)$ by Lemma A.1.3, so $F\sigma_1{\downarrow} = G\sigma_1{\downarrow}$ iff $F\sigma_2{\downarrow} = G\sigma_2{\downarrow}$ by Lemma A.1.2.1.

□

A.2 Proofs of Chapter 4

The relation $>_{\mathrm{a}}$ is a standard labelled bisimulation.

Lemma A.2.1 (Lemma 4.1.7) *If $P >_{\mathrm{a}} Q$ then*

- *If $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu} Q'$ and $P' >_{\mathrm{a}} Q'$.*
- *If $Q \xrightarrow{\mu} Q'$ such that $\mathrm{bn}(\mu) \cap \mathrm{fn}(P) = \emptyset$ then $P \xrightarrow{\mu} P'$ and $P' >_{\mathrm{a}} Q'$.*

Proof: We assume that $P >_{\mathrm{a}} Q$ and that $P \xrightarrow{\mu} P'$ is derived without using bound names that are free in Q, and conversely for transitions of Q. We proceed by induction on the derivation of $P >_{\mathrm{a}} Q$ and the derivation of transitions.

Certain cases of the inner induction are independent of the outer induction hypothesis for their proof:

out Here $P = \overline{F_1}\langle F_2\rangle.P'$ and $Q = \overline{G_1}\langle G_2\rangle.Q'$ where $F_1 >_{\mathrm{a}} G_1$, $F_2 >_{\mathrm{a}} G_2$ and $P' >_{\mathrm{a}} Q'$.

$P \xrightarrow{\overline{a}\,M} P'$: Here $\mathbf{e}(F_1) = a$ and $\mathbf{e}(F_2) = M$. By Lemma 4.1.6.1, $\mathbf{e}(G_1) = a$ and $\mathbf{e}(G_2) = M$ so $Q \xrightarrow{\overline{a}\,M} Q'$ where $P' >_{\mathrm{a}} Q'$.

$Q \xrightarrow{\overline{a}\,M} Q'$: Here $\mathbf{e}(G_1) = a$ and $\mathbf{e}(G_2) = M$. By Lemma 4.1.6.1, $\mathbf{e}(F_1) = a$ and $\mathbf{e}(F_2) = M$ so $P \xrightarrow{\overline{a}\,M} P'$ where $P' >_{\mathrm{a}} Q'$.

inp Here $P = F(x).P'$ and $Q = G(x).Q'$ where $F >_{\mathrm{a}} G$ and $P' >_{\mathrm{a}} Q'$.

$P \xrightarrow{a\,x} P'$: Here $\mathbf{e}(F) = a$. By Lemma 4.1.6.1, $\mathbf{e}(G) = a$, so $Q \xrightarrow{a\,x} Q'$. Then $P'\{^{M}/_{x}\} >_{\mathrm{a}} Q'\{^{M}/_{x}\}$.

$Q \xrightarrow{a\,x} Q'$: Here $\mathbf{e}(G) = a$. By Lemma 4.1.6.1, $\mathbf{e}(F) = a$, so $P \xrightarrow{a\,x} P'$. Then $P'\{^{M}/_{x}\} >_{\mathrm{a}} Q'\{^{M}/_{x}\}$.

rep Here $P =! F(x).P'$ and $Q =! G(x).Q'$ where $F >_{\mathrm{a}} G$ and $P' >_{\mathrm{a}} Q'$.

$P \xrightarrow{a\,x} P' \mid P$: Here $\mathbf{e}(F) = a$. By Lemma 4.1.6.1, $\mathbf{e}(G) = a$, so $Q \xrightarrow{a\,x} Q' \mid Q$. Then $P'\{^{M}/_{x}\} \mid P >_{\mathrm{a}} Q'\{^{M}/_{x}\} \mid Q$.

$Q \xrightarrow{a\,x} Q' \mid Q$: Here $\mathbf{e}(G) = a$. By Lemma 4.1.6.1, $\mathbf{e}(F) = a$, so $P \xrightarrow{a\,x} P' \mid P$. Then $P'\{^{M}/_{x}\} \mid P >_{\mathrm{a}} Q'\{^{M}/_{x}\} \mid Q$.

alp

$P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} P'$: Here $P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,N} P''$ and $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is bijective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(a, M) \cup \mathrm{fn}(P'')) = \emptyset$, $M = N\sigma$ and $P' = P''\sigma$. By assumption

$(\{\tilde{b}\} \cup \{\tilde{c}\}) \cap \mathrm{fn}(Q) = \emptyset$ so by induction $Q \xrightarrow{(\nu\tilde{b})\,\overline{a}\,N} Q'$ with $P'' >_{\mathrm{a}} Q'$. Since $\mathrm{fn}(Q') \subseteq \mathrm{fn}(Q) \cup \{\tilde{b}\}$, $(Q \xrightarrow{(\nu\tilde{b})\,\overline{a}\,N} Q') =_\alpha (Q \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} Q'\sigma)$. By Definition 4.1.4.1 $P' >_{\mathrm{a}} Q'\sigma$.

$Q \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} Q'$: Here $Q \xrightarrow{(\nu\tilde{b})\,\overline{a}\,N} Q''$ and $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is bijective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(a, M) \cup \mathrm{fn}(Q'')) = \emptyset$, $M = N\sigma$ and $Q' = Q''\sigma$. By assumption $(\{\tilde{b}\} \cup \{\tilde{c}\}) \cap \mathrm{fn}(P) = \emptyset$ so by induction $P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,N} P'$ with $P' >_{\mathrm{a}} Q''$. Since $\mathrm{fn}(P') \subseteq \mathrm{fn}(P) \cup \{\tilde{b}\}$, $(P \xrightarrow{(\nu\tilde{b})\,\overline{a}\,N} P') =_\alpha (P \xrightarrow{(\nu\tilde{c})\,\overline{a}\,M} P'\sigma)$. By Definition 4.1.4.1 $P' >_{\mathrm{a}} Q'\sigma$.

grd

$P = \phi P' \xrightarrow{\mu} P''$: Here $P' \xrightarrow{\mu} P''$, $[\![\phi]\!]$ and $Q = \psi Q'$ with $\phi >_{\mathrm{a}} \psi$ and $P' >_{\mathrm{a}} Q'$. By induction $Q' \xrightarrow{\mu} Q''$ with $P'' >_{\mathrm{a}} Q''$. Then $[\![\psi]\!]$ by Lemma 4.1.6.1, so $\psi Q' \xrightarrow{\mu} Q''$ by GRD.

$Q = \psi Q' \xrightarrow{\mu} Q''$: Here $Q' \xrightarrow{\mu} Q''$, $[\![\psi]\!]$ and $P = \phi P'$ with $\phi >_{\mathrm{a}} \psi$ and $P' >_{\mathrm{a}} Q'$. By induction $P' \xrightarrow{\mu} P''$ with $P'' >_{\mathrm{a}} Q''$. Then $[\![\phi]\!]$ by Lemma 4.1.6.1, so $\psi P' \xrightarrow{\mu} P''$ by GRD.

We now return to the outer induction, where the cases where the last rule used in the derivation of the transition is GRD, INP, OUT, REP or ALP have already been treated.

reflexivity If $P = Q$ then $P \sim Q$.
base cases

1. By the INP, OUT and GRD cases.
2. If $P = (\nu a)\,(\nu b)\,R$ and $Q = (\nu b)\,(\nu a)\,R$, we only consider transitions of P by symmetry.
   (a) If $a = b$ then $P = Q$ and we are in the reflexive case treated above.
   (b) If $P \xrightarrow{\tau} (\nu a)\,(\nu b)\,R'$ by $R \xrightarrow{\tau} R'$ and RES (twice), then $Q \xrightarrow{\tau} (\nu b)\,(\nu a)\,R'$ where $(\nu b)\,(\nu a)\,R' >_{\mathrm{a}} (\nu b)\,(\nu a)\,R'$.
   (c) If $P \xrightarrow{c\,x} (\nu a)\,(\nu b)\,R'$ by $R \xrightarrow{c\,x} R'$ and RES (twice), then $Q \xrightarrow{c\,x} (\nu b)\,(\nu a)\,R'$ where $(\nu b)\,(\nu a)\,R' >_{\mathrm{a}} (\nu b)\,(\nu a)\,R'$.
   (d) For output transitions, we can collapse zero or more sequential uses of the ALP rule to a single use since $=_\alpha$ is an equivalence relation on transitions. The case where ALP is the last rule to be used has been treated above. Then there remain four cases.
   res,res Assume that $R \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R'$ with $b \notin \mathrm{n}(c, M)$, that $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is injective with $a \notin \mathrm{n}(c, \mathbf{e}(M\sigma))$ and $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R')) = \emptyset$, so $P \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(((\nu b)\,R')\sigma)$. By assumption,

$\{\tilde{b}\} \cup \{\tilde{c}\} \cap \mathrm{fn}(Q) = \emptyset$. By ALP $R \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R'\sigma$. By RES $(\nu a)\,R \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(R'\sigma)$.

If $b \notin \{\tilde{c}\}$ then $b \notin \mathrm{n}(c, \mathbf{e}(M\sigma))$ and $Q \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu b)\,(\nu a)\,(R'\sigma)$ by RES.

If $b \in \{\tilde{c}\}$ then $b \notin \mathrm{fn}(R')$ and we can take $d, e \notin \{\tilde{b}\} \cup \{\tilde{c}\} \cup \{a, b, c\} \cup \mathrm{fn}(P, R')$ and let $\rho = \{b \mapsto d\} \cup \{n \mapsto n \mid n \in \{\tilde{c}\} \setminus \{b\}\}$. Then $(\nu a)\,R \xrightarrow{(\nu\widetilde{c\sigma})\,\overline{c}\,\mathbf{e}(\mathbf{e}(M\sigma)\rho)} ((\nu a)\,R')\sigma\rho$ by ALP.

By RES $(\nu b)\,(\nu a)\,R \xrightarrow{(\nu\widetilde{c\sigma})}\, \overline{c}\;\mathbf{e}(\mathbf{e}(M\sigma)\rho)(\nu b)\,(((\nu a)\,R')\sigma\rho)$, and by ALP $(\nu b)\,(\nu a)\,R \xrightarrow{(\nu\widetilde{c})\,\overline{c}\,\mathbf{e}(\mathbf{e}(\mathbf{e}(M\sigma)\rho)\rho^{-1})} ((\nu b)\,(((\nu a)\,R')\sigma\rho))\rho^{-1}$.

Here $\mathbf{e}(\mathbf{e}(\mathbf{e}(M\sigma)\rho)\rho^{-1}) = \mathbf{e}(\mathbf{e}(M\sigma)) = \mathbf{e}(M\sigma)$, $((\nu b)\,R')\sigma = (\nu e)\,(R'\sigma)$ and $((\nu b)\,(((\nu a)\,R')\sigma\rho))\rho^{-1} = (\nu e)\,(\nu a)\,(R'\sigma)$ where $(\nu a)\,(\nu e)\,(R'\sigma) >_{\mathrm{a}} (\nu e)\,(\nu a)\,(R'\sigma)$.

res,open Assume that $R \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R'$ with $b \notin \mathrm{n}(c, M)$, that $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R')) = \emptyset$ and that $\mathrm{n}(\mathbf{e}(M\sigma)) \ni a \notin \{c, \widetilde{b}\}$, yielding $P \xrightarrow{(\nu a)\,(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} ((\nu b)\,R')\sigma)$. By ALP $R \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R'\sigma$. By OPEN $(\nu a)\,R \xrightarrow{(\nu a,\widetilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R'\sigma$.

If $b \notin \{\tilde{c}\}$ then $b \notin \mathrm{n}(c, \mathbf{e}(M\sigma))$ and $Q \xrightarrow{(\nu a,\widetilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu b)\,(R'\sigma)$ by RES.

If $b \in \{\tilde{c}\}$ then $b \notin \mathrm{fn}(R')$ and we can take $d, e \notin \{\tilde{b}\} \cup \{\tilde{c}\} \cup \{a, b, c\} \cup \mathrm{fn}(P, R')$ and let $\rho = \{b \mapsto d\} \cup \{n \mapsto n \mid n \in \{\tilde{c}\} \setminus \{b\}\}$. Then $(\nu a)\,R \xrightarrow{(\nu a\widetilde{c\sigma})}\, \overline{c}\;\mathbf{e}(\mathbf{e}(M\sigma)\rho)R'\sigma\rho$ by ALP.

By RES $(\nu b)\,(\nu a)\,R \xrightarrow{(\nu a,\widetilde{c\sigma})}\, \overline{c}\;\mathbf{e}(\mathbf{e}(M\sigma)\rho)(\nu b)\,(R'\sigma\rho)$, and by ALP $(\nu b)\,(\nu a)\,R \xrightarrow{(\nu a,\widetilde{c})\,\overline{c}\,\mathbf{e}(\mathbf{e}(\mathbf{e}(M\sigma)\rho)\rho^{-1})} ((\nu b)\,(R'\sigma\rho))\rho^{-1}$.

Here $\mathbf{e}(\mathbf{e}(\mathbf{e}(M\sigma)\rho)\rho^{-1}) = \mathbf{e}(\mathbf{e}(M\sigma)) = \mathbf{e}(M\sigma)$, $((\nu b)\,R')\sigma = (\nu e)\,(R'\sigma)$ and $((\nu b)\,(R'\sigma\rho))\rho^{-1} = (\nu e)\,(R'\sigma)$.

open,res Assume that $R \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R'$ with $\mathrm{n}(M) \ni b \notin \{c, \widetilde{b}\}$, that $\sigma : \{b, \widetilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{b, \widetilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R')) = \emptyset$ and that $a \notin \mathrm{n}(c, \mathbf{e}(M\sigma))$, yielding $P \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(R'\sigma)$.

If $a \in \{\tilde{b}\}$ then $a \notin \mathrm{fn}(R'\sigma)$. We then take $d \notin \{\tilde{b}\} \cup \{\tilde{c}\} \cup \{a, b, c\} \cup \mathrm{fn}(P, R')$ and let $\rho = \{n \mapsto n \mid n \in \{\tilde{b}\} \wedge n \neq a\} \cup \{a \mapsto d\}$. By ALP $R \xrightarrow{(\nu\widetilde{b\rho})\,\overline{c}\,\mathbf{e}(M\rho)} R'\rho$. By RES $(\nu a)\,R \xrightarrow{(\nu\widetilde{b\rho})\,\overline{c}\,\mathbf{e}(M\rho)} (\nu a)\,(R'\rho)$. By OPEN $(\nu b)\,(\nu a)\,R \xrightarrow{(\nu b)\,(\nu\widetilde{b\rho})\,\overline{c}\,\mathbf{e}(M\rho)} (\nu a)\,(R'\rho)$. Let $\sigma' = \{n \mapsto \sigma(n) \mid n \in \{\tilde{b}, b\} \wedge n \neq a\} \cup \{d \mapsto \sigma(a)\}$. By ALP

$(\nu b)\,(\nu a)\,R \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} ((\nu a)\,(R'\rho))\rho^{-1}\sigma$. Here $((\nu a)\,(R'\rho))\rho^{-1}\sigma = ((\nu d)\,R')\sigma = (\nu a)\,(R'\sigma)$.

open,open Assume that $R \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R'$ with $\mathrm{n}(M) \ni b \notin \{c, \widetilde{b}\}$, that $\sigma : \{b, \widetilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{b, \widetilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R')) = \emptyset$ and that $\mathrm{n}(\mathbf{e}(M\sigma)) \ni a \notin \{c, \widetilde{c}\}$, yielding $P \xrightarrow{(\nu a)\,(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R'\sigma$.

Take $d \notin \{\tilde{b}\} \cup \{\tilde{c}\} \cup \{a, b, c\} \cup \mathrm{fn}(P, R')$ and let $\rho = \{n \mapsto n \mid n \in \{\tilde{b}\} \wedge n \neq a\} \cup \{n \mapsto d \mid n \in \{\tilde{b}\} \wedge n = a\}$. By ALP $R \xrightarrow{(\nu\widetilde{b\rho})\,\overline{c}\,\mathbf{e}(M\rho)} R'\rho$.

By OPEN $(\nu a)\,R \xrightarrow{(\nu a)\,(\nu\widetilde{b\rho})\,\overline{c}\,\mathbf{e}(M\rho)} R'\rho$. Agan by OPEN $(\nu b)\,(\nu a)\,R \xrightarrow{(\nu b)\,(\nu a)\,(\nu\widetilde{b\rho})\,\overline{c}\,\mathbf{e}(M\rho)} R'\rho$. Let $\sigma' = \{a \mapsto a\} \cup \{n \mapsto \sigma(n) \mid n \in \{\tilde{b}, b\} \wedge n \neq a\} \cup \{d \mapsto \sigma(a) \mid a \in \{\tilde{b}\}\}$ and $\widetilde{e} = a\widetilde{c}$. By ALP $(\nu b)\,(\nu a)\,R \xrightarrow{(\nu\tilde{e})\,\overline{c}\,\mathbf{e}(M\sigma)} R'\sigma$.

3. Here $P = (\nu a)\,(R_1 \mid R_2)$ and $Q = R_1 \mid (\nu a)\,R_2$ where $a \notin \mathrm{fn}(R_1)$. As above, for output transitions we consider only derivations with exactly one application of ALP between the uses of any two other rules. There are the following cases for transitions of P.

par-l,res Assume that $P \xrightarrow{\mu} (\nu a)\,(R_1' \mid R_2)$ by $R_1 \xrightarrow{\mu} R_1'$ and $\mu = \tau$ or $\mu = c\,x$. Then $c \neq a$ so $Q \xrightarrow{\mu} R_1' \mid (\nu a)\,R_2$ by PAR-L.

par-l,alp,res Assume that $P \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(R_1' \mid R_2)$ by $R_1 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R_1'$, $\{\tilde{b}\} \cap \mathrm{fn}(R_2) = \emptyset$ and that $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R_1' \mid R_2)) = \emptyset$ and $a \notin \mathrm{n}(c, \mathbf{e}(M\sigma))$.

Then $Q \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R_1' \mid (\nu a)\,R_2$ by PAR-L,ALP.

par-r,res Assume that $P \xrightarrow{\mu} (\nu a)\,(R_1 \mid R_2')$ by $R_2 \xrightarrow{\mu} R_2'$ and $\mu = \tau$ or $\mu = c\,x$ with $c \neq a$. Then $Q \xrightarrow{\mu} R_1 \mid (\nu a)\,R_2'$ by RES,PAR-R.

par-r,alp,res Assume that $P \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(R_1 \mid R_2')$ by $R_2 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R_2'$, $\{\tilde{b}\} \cap \mathrm{fn}(R_1) = \emptyset$ and that $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R_1 \mid R_2')) = \emptyset$ and $a \notin \mathrm{n}(c, \mathbf{e}(M\sigma))$.

Then $Q \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R_1 \mid (\nu a)\,R_2'$ by ALP,RES,PAR-R.

par-r,alp,open Assume that $P \xrightarrow{(\nu a)\,(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R_1 \mid R_2'$ by $R_2 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R_2'$, $\{\tilde{b}\} \cap \mathrm{fn}(R_1) = \emptyset$ and that $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R_1 \mid R_2')) = \emptyset$ and $\mathrm{n}(\mathbf{e}(M\sigma)) \ni a \notin \mathrm{n}(c, \widetilde{c})$.

Then $Q \xrightarrow{(\nu a)\,(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R_1 \mid R_2'$ by ALP,OPEN,PAR-L.

com-l,res Assume that $P \xrightarrow{\tau} (\nu a)\,(\nu\tilde{b})\,(R_1' \mid R_2'\{^{M}/_{x}\})$ by $R_1 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R_1'$ and $R_2 \xrightarrow{c\,x} R_2'$ with $\{\tilde{b}\} \cap \mathrm{fn}(R_2') = \emptyset$ and $a \notin \mathrm{fn}(R_1) \ni c$.

If $a \notin \{\tilde{b}\}$ then we get $Q \xrightarrow{\tau} (\nu\tilde{b})\,(R_1' \mid (\nu a)\,(R_2'\{^M/_x\}))$ by RES,COM-L, where $(\nu a)\,(\nu\tilde{b})\,(R_1' \mid R_2'\{^M/_x\})) >_{\mathrm{a}} (\nu\tilde{b})\,(R_1' \mid (\nu a)\,(R_2'\{^M/_x\}))$.
If $a \in \{\tilde{b}\}$ then $a \notin \mathrm{fn}(R_2')$. We take $d \notin \{\tilde{b}\} \cup \mathrm{n}(a, M) \cup \mathrm{fn}(R_1', R_2')$. By RES,COM-L $Q \xrightarrow{\tau} (\nu\tilde{b})\,(R_1' \mid (\nu d)\,(R_2'\{^M/_x\}))$ where
$(\nu a)\,(\nu\tilde{b})\,(R_1' \mid R_2'\{^M/_x\})) =_\alpha (\nu d)\,(\nu\tilde{b})\,(R_1' \mid R_2'\{^M/_x\})) >_{\mathrm{a}}$
$(\nu\tilde{b})\,(R_1' \mid (\nu d)\,(R_2'\{^M/_x\}))$.

com-r,res Assume that $P \xrightarrow{\tau} (\nu a)\,(\nu\tilde{b})\,(R_1'\{^M/_x\} \mid R_2')$ by $R_2 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R_2'$ and $R_1 \xrightarrow{c\,x} R_1'$ with $\{\tilde{b}\} \cap \mathrm{fn}(R_1') = \emptyset$ and $a \notin \mathrm{fn}(R_1) \ni c$.
If $a \notin \{\tilde{b}\} \cup \mathrm{n}(M)$ then we get $Q \xrightarrow{\tau} (\nu\tilde{b})\,(R_1'\{^M/_x\} \mid (\nu a)\,R_2')$ by RES,COM-L, where
$(\nu a)\,(\nu\tilde{b})\,(R_1' \mid R_2'\{^M/_x\})) >_{\mathrm{a}} (\nu\tilde{b})\,(R_1' \mid (\nu a)\,(R_2'\{^M/_x\}))$.
If $a \in \mathrm{n}(M) \setminus \{\tilde{b}\}$ then we get $Q \xrightarrow{\tau} (\nu a)\,(\nu\tilde{b})\,(R_1'\{^M/_x\} \mid R_2')$ by OPEN,COM-L.
If $a \in \{\tilde{b}\}$ then we take $d \notin \{\tilde{b}\} \cup \mathrm{n}(a, M) \cup \mathrm{fn}(R_1', R_2')$ and let $\sigma = \{a \mapsto d\} \cup \{n \mapsto n \mid n \in \{\tilde{b}\} \wedge n \neq a\}$. By ALP and RES $(\nu a)\,R_2 \xrightarrow{(\nu\tilde{b}sigma)\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(R_2'\sigma)$. $(\nu a)\,R_2 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} (\nu d)\,R_2'$ by ALP, and by COM-R $Q \xrightarrow{\tau} (\nu\tilde{b})\,(R_1'\{^M/_x\} \mid (\nu d)\,R_2')$ where
$(\nu a)\,(\nu\tilde{b})\,(R_1'\{^M/_x\} \mid R_2')) =_\alpha (\nu d)\,(\nu\tilde{b})\,(R_1'\{^M/_x\} \mid R_2')) >_{\mathrm{a}}$
$(\nu\tilde{b})\,(R_1' \mid (\nu d)\,(R_2'\{^M/_x\}))$.

There are the following cases for transitions of Q.

par-l Assume that $Q \xrightarrow{\mu} R_1' \mid (\nu a)\,R_2$ by $R_1 \xrightarrow{\mu} R_1'$ and $\mathrm{bn}(\mu) \cap \mathrm{fn}((\nu a)\,R_2) = \emptyset$. If $a \notin \mathrm{bn}(\mu)$ then $P \xrightarrow{\mu} (\nu a)\,(R_1' \mid R_2)$ by PAR-L,RES.
If $\mu = (\nu\tilde{b})\,\overline{c}\,M$ and $a \in \{\tilde{b}\}$ then we take $d \notin \{\tilde{b}\} \cup \mathrm{n}(a, M) \cup \mathrm{fn}(R_1', R_2)$ and let $\sigma = \{a \mapsto d\} \cup \{n \mapsto n \mid n \in \{\tilde{b}\} \wedge n \neq a\}$. By ALP and PAR-L $P \xrightarrow{(\nu\tilde{b}\sigma)\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(R_1'\sigma \mid R_2)$. $P \xrightarrow{(\nu\tilde{b})\,\overline{c}\,\mathbf{e}(M)}$ $(\nu d)\,(R_1' \mid R_2\sigma)$ by ALP.

res,par-r Assume that $Q \xrightarrow{\mu} R_1 \mid (\nu a)\,R_2'$ by $R_2 \xrightarrow{\mu} R_2'$ and $\mu = \tau$ or $\mu = c\,x$ with $c \neq a$. Then $P \xrightarrow{\mu} (\nu a)\,(R_1 \mid R_2')$ by PAR-R,RES.

res,alp,par-r Assume that $Q \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R_1 \mid (\nu a)\,R_2'$ by $R_2 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M} R_2'$, $a \notin \mathrm{n}(c, M)$ and that $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R_2')) = \emptyset$ and $\{\tilde{c}\} \cap \mathrm{fn}(R_1) = \emptyset$. By assumption $\{\tilde{b}\} \cap \mathrm{fn}(R_1) = \emptyset$.
Then $P \xrightarrow{(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} (\nu a)\,(R_1 \mid R_2')$ by PAR-R,RES,ALP.

open,alp,par-r Assume that $Q \xrightarrow{(\nu a)\,(\nu\tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R_1 \mid R_2'$ by $R_2 \xrightarrow{(\nu\tilde{b})\,\overline{c}\,M}$

R'_2, $\mathrm{n}(M) \ni a \notin \{c, \widetilde{b}\}$ and that $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ is injective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R'_2)) = \emptyset$ and $\{\tilde{c}\} \cap \mathrm{fn}(R_1) = \emptyset$. By assumption $\{\tilde{b}\} \cap \mathrm{fn}(R_1) = \emptyset$.

Then $P \xrightarrow{(\nu a)\,(\nu \tilde{c})\,\overline{c}\,\mathbf{e}(M\sigma)} R_1 \mid R'_2$ by PAR-R,OPEN,ALP.

com-l Assume that $Q \xrightarrow{\tau} (\nu \tilde{b})\,(R'_1 \mid (((\nu a)\,R'_2)\{^M/_x\}))$ by $R_1 \xrightarrow{(\nu \tilde{b})\,\overline{c}\,M} R'_1$ and $R_2 \xrightarrow{c\,x} R'_2$ with $\{\tilde{b}\} \cap \mathrm{fn}((\nu a)\,R_2) = \emptyset$ and $c \neq a$.

If $a \notin \{\tilde{b}\}$ then $((\nu a)\,R'_2)\{^M/_x\} = (\nu a)\,(R'_2\{^M/_x\})$, and by COM-L,RES we get $P \xrightarrow{\tau} (\nu a)\,(\nu \tilde{b})\,(R'_1 \mid (R'_2\{^M/_x\})) >_{\mathrm{a}} (\nu \tilde{b})\,(R'_1 \mid (\nu a)\,(R'_2\{^M/_x\}))$.

If $a \in \{\tilde{b}\}$ then we take $d \notin \{\tilde{b}\} \cup \mathrm{n}(a, M) \cup \mathrm{fn}(R'_1, R'_2)$, let $\sigma = \{a \mapsto d, d \mapsto a\}$ and get $((\nu a)\,R'_2)\{^M/_x\} = (\nu d)\,(R'_2\sigma\{^M/_x\})$. By ALP,COM-L,RES we get

$$\begin{aligned} P &\xrightarrow{\tau} (\nu a)\,(\nu \tilde{b}\sigma)\,(R'_1\sigma \mid (R'_2\{^{\mathbf{e}(M\sigma)}/_x\})) \\ &=_{\alpha} (\nu d)\,(\nu \tilde{b})\,(R'_1 \mid (R'_2\{^{\mathbf{e}(M\sigma)}/_x\}\sigma)) \\ &>_{\mathrm{a}} (\nu \tilde{b})\,(R'_1 \mid (\nu d)\,(R'_2\sigma\{^M/_x\})) \end{aligned}$$

res,alp,com-r Assume that $Q \xrightarrow{\tau} (\nu \tilde{c})\,(R'_1\{^{\mathbf{e}(M\sigma)}/_x\} \mid (\nu a)\,R'_2)$ by $R_2 \xrightarrow{(\nu \tilde{b})\,\overline{c}\,M} R'_2$, $R_1 \xrightarrow{c\,x} R'_1$, $a \notin \{\tilde{b}, c\} \cup \mathrm{n}(M)$ and $\sigma : \{\tilde{b}\} \to \{\tilde{c}\}$ bijective with $(\{\tilde{c}\} \setminus \{\tilde{b}\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}((\nu a)\,R'_2)) = \emptyset$ and $\{\tilde{c}\} \cap \mathrm{fn}(R'_1) = \emptyset$.

If $a \notin \{\tilde{c}\}$ we get $P \xrightarrow{\tau} (\nu a)\,(\nu \tilde{c})\,(R'_1\{^{\mathbf{e}(M\sigma)}/_x\} \mid R'_2\sigma)$ by ALPH,COM-R,RES.

If $a \in \{\tilde{c}\}$ then we take $d \notin \{\tilde{b}\} \cup \mathrm{n}(a, M) \cup \mathrm{fn}(R'_1, R'_2)$, and let $\sigma' = \{n \mapsto d \mid n \in \{\tilde{b}\} \wedge \sigma(n) = a\} \cup \{n \mapsto \sigma(n) \mid n \in \{\tilde{b}\} \wedge \sigma(n) \neq a\}$ and $\rho = \{a \mapsto d, d \mapsto a\}$. By ALPH,COM-R,RES

$$\begin{aligned} P &\xrightarrow{\tau} (\nu a)\,(\nu \tilde{b}\sigma')\,(R'_1\{^{\mathbf{e}(M\sigma')}/_x\} \mid R'_2\sigma') \\ &=_{\alpha} (\nu d)\,(\nu \tilde{c})\,(R'_1\{^{\mathbf{e}(M\sigma')\rho}/_x\} \mid (R'_2\sigma'\rho)) \\ &>_{\mathrm{a}} (\nu \tilde{c})\,(R'_1\{^{\mathbf{e}(M\sigma)}/_x\} \mid (\nu d)\,(R'_2\sigma'\rho)) \\ &=_{\alpha} (\nu \tilde{c})\,(R'_1\{^{\mathbf{e}(M\sigma)}/_x\} \mid (\nu a)\,R'_2\sigma) \end{aligned}$$

open,alp,com-r Assume that $Q \xrightarrow{\tau} (\nu \tilde{c})\,(R'_1\{^{\mathbf{e}(M\sigma)}/_x\} \mid R'_2)$ by $R_2 \xrightarrow{(\nu \tilde{b})\,\overline{c}\,M} R'_2$, $R_1 \xrightarrow{c\,x} R'_1$, $\mathrm{n}(M) \ni a \notin \{\tilde{b}, c\}$ and that $\sigma : \{\tilde{b}, a\} \to \{\tilde{c}\}$ is bijective with $(\{\tilde{c}\} \setminus \{\tilde{b}, a\}) \cap (\mathrm{n}(c, M) \cup \mathrm{fn}(R'_2)) = \emptyset$ and $\{\tilde{c}\} \cap \mathrm{fn}(R'_1) = \emptyset$.

We take $d \notin \{\tilde{b}, \widetilde{c}, a\} \cup \mathrm{n}(M) \cup \mathrm{fn}(R'_1, R'_2)$, and let $\sigma' = \{n \mapsto d \mid n \in \{\tilde{b}\} \wedge \sigma(n) = a\} \cup \{n \mapsto \sigma(n) \mid n \in \{\tilde{b}\} \wedge \sigma(n) \neq a\}$ and

$\rho = \{a \mapsto \sigma(a), d \mapsto a\}$. By ALP,COM-R,RES

$$\begin{aligned} P &\xrightarrow{\tau} (\nu a)\,(\nu \tilde{b}\sigma')\,(R_1'\{^{\mathbf{e}(M\sigma')}/_x\} \mid R_2'\sigma') \\ &=_\alpha (\nu \tilde{c})\,(R_1'\{^{\mathbf{e}(M\sigma')\rho}/_x\} \mid (R_2'\sigma'\rho)) \\ &>_\mathrm{a} (\nu \tilde{c})\,(R_1'\{^{\mathbf{e}(M\sigma)}/_x\} \mid (R_2'\sigma)) \end{aligned}$$

4. As 3, by rule symmetry.

context cases The (replicated) input, output and guard prefix cases have already been treated.

$P = P_1 \mid P_2$: In this case, $Q = Q_1 \mid Q_2$ with $P_1 >_\mathrm{a} Q_1$ and $P_2 >_\mathrm{a} Q_2$. By rule symmetry and the prior proof for the ALP rule, we need to consider two cases.

par-l Assume that $P_1 \xrightarrow{\mu} P_1'$ with $\mathrm{bn}(\mu) \cap \mathrm{fn}(P_2) = \emptyset$, yielding $P \xrightarrow{\mu} P_1' \mid P_2$. By assumption $\mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$.
By induction $Q_1 \xrightarrow{\mu} Q_1'$ with $P_1' >_\mathrm{a} Q_1'$, yielding $Q \xrightarrow{\mu} Q_1' \mid Q_2$ with $P_1' \mid P_2 >_\mathrm{a} Q_1' \mid Q_2$.

com-l Assume that $P_1 \xrightarrow{(\nu \tilde{b})\,\overline{a}\,M} P_1'$ and $P_2 \xrightarrow{a\,x} P_2'$ with $\{\tilde{b}\} \cap \mathrm{fn}(P_2') = \emptyset$, yielding $P \xrightarrow{\tau} (\nu \tilde{b})\,(P_1' \mid P_2'\{^M/_x\})$. By assumption $\{\tilde{b}\} \cap \mathrm{fn}(Q) = \emptyset$.
By induction $Q_1 \xrightarrow{(\nu \tilde{b})\,\overline{a}\,M} Q_1'$ with $P_1' >_\mathrm{a} Q_1'$ and $Q_2 \xrightarrow{a\,x} Q_2'$ with $P_2' >_\mathrm{a} Q_2'$, yielding $Q \xrightarrow{\tau} (\nu \tilde{b})\,(Q_1' \mid Q_2'\{^M/_x\})$ where $(\nu \tilde{b})\,(P_1' \mid P_2'\{^M/_x\}) >_\mathrm{a} (\nu \tilde{b})\,(Q_1' \mid Q_2'\{^M/_x\})$.

$P = (\nu a)\,P_1$: In this case, $Q = (\nu a)\,Q_1$ with $P_1 >_\mathrm{a} Q_1$. By induction, if $P_1 \xrightarrow{\mu} P_1'$ then $Q_1 \xrightarrow{\mu} Q_1'$ with $P_1' >_\mathrm{a} Q_1'$ and conversely. Assume that $P \xrightarrow{\mu} P'$; the case where $Q \xrightarrow{\mu} Q'$ is treated in the same way. The last rule used in the derivation is either ALP, OPEN or RES, where the ALP case has already been treated.

res If $a \notin \mathrm{n}(\mu)$ then $P \xrightarrow{\mu} (\nu a)\,P_1'$ and $Q \xrightarrow{\mu} (\nu a)\,Q_1'$.

open If $\mu = (\nu \tilde{b})\,\overline{c}\,M$ with $\mathrm{n}(M) \ni b \notin \{c, \tilde{b}\}$ then $P \xrightarrow{(\nu a)\,(\nu \tilde{b})\,\overline{c}\,M} P_1'$ and $Q \xrightarrow{(\nu a)\,(\nu \tilde{b})\,\overline{c}\,M} Q_1'$.

transitive case In this case, $P >_\mathrm{a} R >_\mathrm{a} Q$.

- Assume that $P \xrightarrow{\mu} P'$. By the outer induction $R \xrightarrow{\mu} R'$ with $P' >_\mathrm{a} R'$. Again, by the outer induction $Q \xrightarrow{\mu} Q'$ with $R' >_\mathrm{a} Q'$. By transitivity $P' >_\mathrm{a} Q'$.

- Assume that $Q \xrightarrow{\mu} Q'$ with $\mathrm{bn}(\mu) \cap \mathrm{fn}(P) = \emptyset$. By Lemma 4.1.6.3 $\mathrm{bn}(\mu) \cap \mathrm{fn}(R) = \emptyset$. By the outer induction $R \xrightarrow{\mu} R'$ with $R' >_\mathrm{a} Q'$. Again, by the outer induction $P \xrightarrow{\mu} P'$ with $P' >_\mathrm{a} R'$. By transitivity $P' >_\mathrm{a} Q'$.

□

Lemma A.2.2 (Lemma 4.1.11)

1. *If* $P \xrightarrow[\phi]{\mu_s} P_1$ *and* σ *is idempotent and satisfies* $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$, $[\![\phi\sigma]\!]$ *and* $\mathrm{n}(\mathbf{e}(F\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ *whenever* $F \in \mathrm{ch}(\mu_s)$ *then there are* $\widetilde{c}, P_2$ *with* $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu)_s\sigma)\} \uplus \{\widetilde{c}\}$ *and* $P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$ *with* $(\nu\tilde{c})\, P_1\sigma >_\mathrm{a} P_2$.
2. *If* σ *is idempotent and* $P\sigma \xrightarrow{\mu} P_1$ *with* $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu) = \emptyset$ *then there are* $\widetilde{c}, P_2, \phi, \mu_s$ *such that* $\mathrm{bn}(\mu_s) = \{\widetilde{c}\} \uplus \mathrm{bn}(\mu)$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$, $P \xrightarrow[\phi]{\mu_s} P_2$ *and* $(\nu\tilde{c})\, P_2\sigma >_\mathrm{a} P_1$.

Proof:

1. Proof by induction on the derivation of the transition. We first α-rename P to ensure that no names in $\mathrm{n}(\mathrm{range}(\sigma))$ or variables in $\mathrm{dom}(\sigma)$ are bound.

   Sout Assume that $\overline{G}\langle F\rangle.P \xrightarrow[[G:\mathcal{N}]\wedge[F:\mathcal{M}]]{\overline{G}\,F} P$, and that $[\![([G:\mathcal{N}] \wedge [F:\mathcal{M}])\sigma]\!]$.
   Thus, $\mathbf{e}(G\sigma) \in \mathcal{N} \subset \mathcal{M}$ and $\mathbf{e}(F\sigma \in \mathcal{M})$.
   We then get $\overline{G\sigma}\langle F\sigma\rangle.P\sigma \xrightarrow{\overline{a}\,M} P\sigma$ by out.

   Sinp Assume that $G(x).P \xrightarrow[[G:\mathcal{N}]]{G(x)} P$, and that $[\![[G\sigma:\mathcal{N}]]\!]$. Then there is $a \in \mathcal{N}$ such that $a = \mathbf{e}(G\sigma)$. Applying inp we get $G\sigma(x).P\sigma \xrightarrow{a\,x} P\sigma$.

   Sguard Assume that $\phi' P \xrightarrow[\phi\wedge\phi']{\mu_s} P_1$, $\mathrm{bn}(\mu_s) \cap \mathrm{n}(\phi') = \emptyset$, $[\![(\phi \wedge \phi')\sigma]\!]$, $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ and $\mathrm{n}(\mathbf{e}(F\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ whenever $F \in \mathrm{ch}(\mu_s)$.
   By induction there are $\widetilde{c}, P_2$ with $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu_s\sigma))\} \uplus \{\widetilde{c}\}$ and $P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$ with $(\nu\tilde{c})\, P_1\sigma >_\mathrm{a} P_2$.
   Since $[\![\phi'\sigma]\!]$ we may apply grd, yielding $\phi'\sigma P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$.

   Scom Assume that $P \mid Q \xrightarrow[\phi_1\wedge\phi_2\wedge[G=G']]{(\nu\tilde{b}\tilde{c})\ \tau} P'\{^F/_x\} \mid Q'$, $[\![(\phi_1 \wedge \phi_2 \wedge [G = G'])\sigma]\!]$ and $\mathrm{n}(\mathrm{range}(\sigma)) \cap \{\widetilde{b}\widetilde{c}\} = \emptyset$.
   We then have $\{\tilde{b}\} \cap \mathrm{fn}(P) = \{\tilde{c}\} \cap \mathrm{fn}(Q) = \{\tilde{c}\} \cap \{\tilde{b}\} = \emptyset$, $P \xrightarrow[\phi_1]{(\nu\tilde{c})\,G(x)} P'$ and $Q \xrightarrow[\phi_2]{(\nu\tilde{b})\,\overline{G'}\,F} Q'$.
   By $[\![(\phi_1 \wedge \phi_2 \wedge [G = G'])\sigma]\!]$, we have $[\![[G\sigma:\mathcal{N}]]\!]$, $[\![[G'\sigma:\mathcal{N}]]\!]$ and $[\![[G = G']\sigma]\!]$. Thus there is a such that $\mathbf{e}(G\sigma) = a = \mathbf{e}(G'\sigma)$, and particularly

$a \in \mathrm{n}(G\sigma) \cap \mathrm{n}(G'\sigma) \subseteq \mathrm{n}(\mathrm{range}(\sigma)) \cup (\mathrm{n}(G) \cap \mathrm{n}(G'))$. Thus $a \notin \{\widetilde{b}\widetilde{c}\}$, so by induction there are $\widetilde{b_1}, \widetilde{b_2}, P_2, Q_2$ with $\{\widetilde{b}\} = \{\widetilde{b_1}\} \uplus \{\widetilde{b_2}\}$, $P\sigma \xrightarrow{a(x)} P_2$ and $Q\sigma \xrightarrow{(\nu\widetilde{b_1})\,\overline{a}\,M} Q_2$ with $(\nu\widetilde{c})\,P'\sigma >_{\mathrm{a}} P_2$ and $(\nu\widetilde{b_2})\,Q'\sigma >_{\mathrm{a}} Q_2$.
By COM we then have $P\sigma \mid Q\sigma \xrightarrow{\tau} (\nu\widetilde{b_1})\,(P_2\{^M/_x\} \mid Q_2)$, where

$$\begin{aligned}(\nu\widetilde{c})\,P'\{^F/_x\}\sigma &= (\nu\widetilde{c})\,P'\sigma\{^{F\sigma}/_x\}\\ &>_{\mathrm{a}} P_2\{^M/_x\}(\nu\widetilde{b_1}\widetilde{b_2}\widetilde{c})\,(P'\{^F/_x\}\sigma \mid Q'\sigma)\\ &>_{\mathrm{a}} (\nu\widetilde{b_1}\widetilde{b_2})\,(((\nu\widetilde{c})\,P'\{^F/_x\}\sigma) \mid Q'\sigma)\\ &>_{\mathrm{a}} (\nu\widetilde{b_1})\,(P_2\{^M/_x\} \mid (\nu\widetilde{b_2})\,Q'\sigma)\\ &>_{\mathrm{a}} (\nu\widetilde{b_1})\,(P_2\{^M/_x\} \mid Q_2).\end{aligned}$$

SOPEN Assume that $(\nu a)\,P \xrightarrow[\phi]{(\nu a)\,\mu_s} P_1$, $\mathrm{n}(\mathrm{range}(\sigma)) \cap (\{a\} \cup \mathrm{bn}(\mu_s)) = \emptyset$, $[\![\phi\sigma]\!]$ and $\mathrm{n}(\mathbf{e}(F\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ whenever $F \in \mathrm{ch}(\mu_s)$. By the side conditions on the transition we get $(\mathrm{fn}(\mu_s) \cup \mathrm{n}(\phi)) \ni a \notin \mathrm{bn}(\mu_s)$.
By induction there are $\widetilde{c}, P_2$ with $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu_s\sigma))\} \uplus \{\widetilde{c}\}$ and $P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$ with $(\nu\tilde{c})\,P_1\sigma >_{\mathrm{a}} P_2$.

If $a \notin \mathrm{n}(\mathbf{e}(\mu_s\sigma))$ we use RES, yielding $(\nu a)\,P\sigma \xrightarrow{(\nu\widetilde{b})\ \mathbf{e}(\mu_s\sigma)} (\nu a)\,P_2$. By congruence $(\nu a)\,(\nu\tilde{c})\,P_1\sigma >_{\mathrm{a}} (\nu a)\,P_2$. Moreover, $a \notin \mathrm{n}(\mathbf{e}(F\sigma)) \subseteq \mathrm{n}(F) \cup \mathrm{n}(\mathrm{range}(\sigma))$.
Otherwise, $\mu_s = (\nu\widetilde{e})\,\overline{G}\,F$ for some $\widetilde{e}, F, G$ and Then $\mathbf{e}(\mu_s\sigma) = \widetilde{b}\overline{d}\,M$ with $a \in \mathrm{n}(M)$ and $a \notin \{\widetilde{b}, d\}$ since $d = \mathbf{e}(F\sigma)$ and $a \notin \mathrm{n}(\mathbf{e}(F\sigma))$ by assumption. We use OPEN to derive $(\nu a)\,P\sigma \xrightarrow{(\nu a\widetilde{b})\ \mathbf{e}(\mu_s\sigma)} P_2$.

SRES Assume that $(\nu a)\,P \xrightarrow[\phi]{\mu_s} (\nu a)\,P_1$, $[\![\phi\sigma]\!]$, $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ and $\mathrm{n}(\mathbf{e}(F\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ whenever $F \in \mathrm{ch}(\mu_s)$.
By assumption $a \notin \mathrm{n}(\mathrm{range}(\sigma))$. By the side conditions on the transition we get $a \notin \mathrm{n}(\mu_s) \cup \mathrm{n}(\phi)$.

Then $a \notin \mathrm{n}(\mathbf{e}(\mu_s\sigma))$, and RES yields $(\nu a)\,P\sigma \xrightarrow{(\nu\widetilde{b})\ \mathbf{e}(\mu_s\sigma)} (\nu a)\,P_2$. By congruence $(\nu a)\,(\nu\tilde{c})\,P_1\sigma >_{\mathrm{a}} (\nu a)\,P_2$. Moreover, $a \notin \mathrm{n}(\mathbf{e}(F\sigma)) \subseteq \mathrm{n}(F) \cup \mathrm{n}(\mathrm{range}(\sigma))$.

SPAR Take $P \mid Q$ and assume that $P \xrightarrow[\phi]{\mu_s} P'$ with $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$, $[\![\phi\sigma]\!]$ and $\mathrm{n}(\mathbf{e}(F\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ whenever $F \in \mathrm{ch}(\mu_s)$, and that $\mathrm{bn}(\mu_s) \cap \mathrm{fn}(Q) = \emptyset = \mathrm{bv}(\mu_s) \cap \mathrm{fv}(Q)$.

By induction there are $\widetilde{c}, P_2$ with $\mathrm{bn}(\mu_s) = \mathrm{bn}(\mathbf{e}(\mu_s\sigma)) \uplus \{\widetilde{c}\}$ and $P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$ with $(\nu\tilde{c})\,P_1\sigma >_{\mathrm{a}} P_2$.

Since $\mathrm{fn}(Q\sigma) \subseteq \mathrm{n}(\mathrm{range}(\sigma)) \cup \mathrm{fn}(Q)$ we have $\mathrm{bn}(\mathbf{e}(\mu_s\sigma)) \cap \mathrm{fn}(Q\sigma) = \emptyset$, so we may apply PAR.

SSUM Take $P + Q$ and assume that $P \xrightarrow[\phi]{\mu_s} P'$ with $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$, $[\![\phi\sigma]\!]$ and $\mathrm{n}(\mathbf{e}(F\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ whenever $F \in \mathrm{ch}(\mu_s)$.
By induction there are $\widetilde{c}, P_2$ with $\mathrm{bn}(\mu_s) = \mathrm{bn}(\mathbf{e}(\mu_s\sigma)) \uplus \{\widetilde{c}\}$ and $P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$ with $(\nu\tilde{c})\, P_1\sigma >_\mathrm{a} P_2$. We may then apply SUM.

SREP Assume that $!\,G(x).P \xrightarrow[[\,G:\mathcal{N}\,]]{G(x)} P \mid !\,G(x).P$ and $[\![[\,G\sigma:\mathcal{N}\,]]\!]$. Then $\exists a \in \mathcal{N}$ such that $a = \mathbf{e}(G\sigma)$. REP yields $G\sigma(x).P\sigma \xrightarrow{a\,x} P\sigma$.

SALP Assume that $P \xrightarrow[\phi]{\mu_s} P'$ with $\mathrm{n}(\mathrm{range}(\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$, $[\![\phi\sigma]\!]$ and $\mathrm{n}(\mathbf{e}(F\sigma)) \cap \mathrm{bn}(\mu_s) = \emptyset$ whenever $F \in \mathrm{ch}(\mu_s)$, and that $\rho : \mathrm{bn}(\mu_s) \to \mathcal{N}$ is injective such that $(\mathrm{range}(\rho) \setminus \mathrm{dom}(\rho)) \cap (\mathrm{n}(\phi) \cup \mathrm{n}(\mathrm{range}(\sigma)) \cup \mathrm{fn}(P')) = \emptyset$.
Then $P \xrightarrow[\phi\rho]{\mu_s\rho} P'\rho$.
By induction there are $\widetilde{c}, P_2$ with $\mathrm{bn}(\mu_s) = \{\mathrm{bn}(\mathbf{e}(\mu_s\sigma))\} \uplus \{\widetilde{c}\}$ and $P\sigma \xrightarrow{\mathbf{e}(\mu_s\sigma)} P_2$ with $(\nu\tilde{c})\, P_1\sigma >_\mathrm{a} P_2$. If $\{\tilde{b}\} \neq \emptyset$, there are $\widetilde{b}, a, M$ such that $\mathbf{e}(\mu_s\sigma) = (\nu\tilde{b})\,\overline{a}\,M$, and we let $\rho' = \{n \mapsto \rho(n) \mid n \in \{\tilde{b}\}\}$. Then $P\sigma \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P_2 \equiv_\alpha P\sigma \xrightarrow{(\nu\widetilde{b\rho'})\,\overline{a}\,M\rho'} P_2\rho'$, so by ALP $P\sigma \xrightarrow{(\nu\widetilde{b\rho'})\,\overline{a}\,M\rho'} P_2\rho'$. Moreover, $(\nu\widetilde{c\rho})\, P_1\sigma\rho >_\mathrm{a} P_2\rho'$.

2. Proof by induction on the derivation of the transition. We first α-rename $P\sigma$ to ensure that no names in $\mathrm{n}(\mathrm{range}(\sigma))$ or variables in $\mathrm{dom}(\sigma)$ are bound.

OUT Assume that $\overline{G\sigma}\langle F\sigma\rangle.P\sigma \xrightarrow{\overline{a}\,M} P\sigma$ with $\mathbf{e}(G\sigma) = a$ and $\mathbf{e}(F\sigma) = M$, so $[\![([\,G:\mathcal{N}\,] \wedge [\,F:\mathcal{M}\,])\sigma]\!]$. SOUT yields $\overline{G}\langle F\rangle.P \xrightarrow[[\,G:\mathcal{N}\,]\wedge[\,F:\mathcal{M}\,]]{\overline{G}\,F} P$. By reflexivity $P\sigma >_\mathrm{a} P\sigma$.

INP Assume that $G\sigma(x).P\sigma \xrightarrow[[\,G\sigma:\mathcal{N}\,]]{a(x)} P\sigma$. Then $\mathbf{e}(G\sigma) = a$, so $[\![[\,G\sigma:\mathcal{N}\,]]\!]$.
By SIN we have $G(x).P \xrightarrow[[\,G:\mathcal{N}\,]]{G\,x} P$. By reflexivity $P\sigma >_\mathrm{a} P\sigma$.

GUARD Assume that $\phi\sigma P\sigma \xrightarrow{\mu} P_1$ where $[\![\phi\sigma]\!]$. By induction there are $\widetilde{c}, P_2, \phi', \mu_s$ such that $\mathrm{bn}(\mu_s) = \{\widetilde{c}\} \uplus \mathrm{bn}(\mu)$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi'\sigma]\!]$ and $P \xrightarrow[\phi']{\mu_s} P_2$ where $(\nu\tilde{c})\, P_2\sigma >_\mathrm{a} P_1$. We use SALP to ensure that $\mathrm{bn}(\mu_s) \cap \mathrm{n}(\phi) = \emptyset$, allowing us to apply SGUARD. We then have $\phi P \xrightarrow[\phi\wedge\phi']{\mu_s} P_2$.

COM Assume that $P\sigma \mid Q\sigma \xrightarrow{\tau} (\nu\tilde{b})\,(P_1\{{}^{M}/_{x}\} \mid Q_1)$, i.e., that there is a with $\{\tilde{b}\} \cap \mathrm{fn}(P\sigma) = \emptyset$, $P \xrightarrow{a(x)} P_1\sigma$ and $Q \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} Q_1\sigma$.
By induction there are $\widetilde{b_1}, \widetilde{b_2}, \widetilde{c}, F, G, G', P_2, Q_2, \phi_1, \phi_2$ such that $\{\widetilde{b_2}\} =$

$\{\widetilde{c}\}\uplus\{\widetilde{b}\}$, $\mathbf{e}(G\sigma) = a = \mathbf{e}(G'\sigma)$, $\mathbf{e}(F\sigma) = M$, $[\![\phi_1\sigma]\!]$, $[\![\phi_2\sigma]\!]$, and $P \xrightarrow[\phi_1]{(\nu\widetilde{b_1})\,G'\,x}$ P_2 and $Q \xrightarrow[\phi_2]{(\nu\widetilde{b_2})\,\overline{G}\,F} Q_2$ with $(\nu\widetilde{b_1})\,P_2\sigma >_{\mathrm{a}} P_1$ and $(\nu\tilde{c})\,Q_2\sigma >_{\mathrm{a}} Q_1$. We use Salp to ensure that $\{\tilde{b_1}\}\cap\mathrm{fn}(Q) = \{\tilde{c}\}\cap\mathrm{fn}(P) = \{\tilde{b_1}\}\cap\{\tilde{b_2}\} = \emptyset$ in order to apply Scom. We then have $P \mid Q \xrightarrow[\phi_1\wedge\phi_2\wedge[\,G=G'\,]]{(\nu\widetilde{b_1}\widetilde{b_2})\,\tau} P_2\{{}^{F}/_{x}\} \mid Q_2$. Finally,

$$\begin{aligned}(\nu\widetilde{b_1}\widetilde{b_2})\,(P_2\{{}^{F}/_{x}\}\sigma \mid Q_2\sigma) &>_{\mathrm{a}} (\nu\widetilde{b}\widetilde{c})\,(((\nu\widetilde{b_2})\,P_2\{{}^{F}/_{x}\}\sigma) \mid Q_2\sigma)\\ &= (\nu\widetilde{b}\widetilde{c})\,(((\nu\widetilde{b_2})\,P_2\sigma\{{}^{F\sigma}/_{x}\}) \mid Q_2\sigma)\\ &>_{\mathrm{a}} (\nu\widetilde{b}\widetilde{c})\,(P_1\{{}^{M}/_{x}\} \mid Q_2\sigma)\\ &>_{\mathrm{a}} (\nu\widetilde{b})\,(P_1\{{}^{M}/_{x}\} \mid (\nu\widetilde{c})\,Q_2\sigma)\\ &>_{\mathrm{a}} (\nu\widetilde{b})\,(P_1\{{}^{M}/_{x}\} \mid Q_1)\end{aligned}$$

OPEN Assume that $(\nu n)\,P\sigma \xrightarrow{(\nu n\tilde{b})\,\overline{a}\,M} P_1$ where $\mathrm{n}(M) \ni n \notin \{a\}\cup\{\tilde{b}\}$.
By induction, there are $F, G, \widetilde{d}, \widetilde{c}, P_2, \phi$ such that $\{\widetilde{d}\} = \{\widetilde{c}\}\uplus\{\widetilde{b}\}$, $\mathbf{e}(F) = M$, $\mathbf{e}(G) = a$, $\{\widetilde{b}\} = \{\tilde{d}\}\cap\mathrm{n}(\mathbf{e}(F\Sigma))$, $[\![\phi\sigma]\!]$ and $P \xrightarrow[\phi]{(\nu\widetilde{d})\,\overline{G}\,F} P_2$ with $(\nu\tilde{c})\,P_2\sigma >_{\mathrm{a}} P_1$. We apply Salp to ensure that $n \notin \{\widetilde{d}\}$. By assumption $n \notin \mathrm{n}(\mathrm{range}(\sigma))$, so $n \in \mathrm{n}(F)$, and we can apply Sopen.

RES Assume that $(\nu n)\,P\sigma \xrightarrow{\mu} (\nu n)\,P_1$ since $n \notin \mathrm{n}(\mu)$. there are $\widetilde{c}, P_2, \phi, \mu_s$ such that $\mathrm{bn}(\mu_s) = \{\widetilde{c}\}\uplus\mathrm{bn}(\mu)$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$ and $P \xrightarrow[\phi]{\mu_s} P_2$ with $(\nu\tilde{c})\,P_2\sigma >_{\mathrm{a}} P_1$. We apply Salp to ensure that $n \notin \mathrm{bn}(\mu_s)$.
If $n \notin \mathrm{n}(\mu_s)$, Sres yields that $(\nu n)\,P \xrightarrow[\phi]{\mu_s} (\nu n)\,P_2$ where $(\nu\tilde{c})\,(\nu n)\,P_2\sigma >_{\mathrm{a}} (\nu n)\,(\nu\tilde{c})\,P_2\sigma >_{\mathrm{a}} P_1$.
If $n \in \mathrm{n}(\mu_s)$, Sopen yields that $(\nu n)\,P \xrightarrow[\phi]{(\nu n)\,\mu_s} P_2$ where $(\nu n)\,(\nu\tilde{c})\,P_2\sigma >_{\mathrm{a}} (\nu n)\,P_1$. Moreover, if $\mu_s = (\nu\widetilde{d})\,\overline{G}\,F$ then $n \notin \mathrm{n}(M) = \mathrm{n}(\mathbf{e}(F\sigma))$.

PAR Assume that $P \mid Q \xrightarrow{\mu} P_1 \mid Q$. By induction there are $\widetilde{c}, P_2, \phi, \mu_s$ such that $\mathrm{bn}(\mu_s) = \{\widetilde{c}\}\uplus\mathrm{bn}(\mu)$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$ and $P \xrightarrow[\phi]{\mu_s} P_2$ where $(\nu\tilde{c})\,P_2\sigma >_{\mathrm{a}} P_1$. The side conditions for Spar are satisfied if $\mathrm{bn}(\mu_s)\cap\mathrm{fn}(Q) = \emptyset$, which can be assured using Salp.

SUM Assume that $P\sigma + Q\sigma \xrightarrow{\mu} P_1$. By induction there are $\widetilde{c}, P_2, \phi, \mu_s$ such that $\mathrm{bn}(\mu_s) = \{\widetilde{c}\}\uplus\mathrm{bn}(\mu)$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$ and $P \xrightarrow[\phi]{\mu_s} P_2$ where $P_2\sigma >_{\mathrm{a}} P_1$.
By Ssum $P + Q \xrightarrow[\phi]{\mu_s} P_2$.

REP Assume that $!\,G\sigma(x).P\sigma \xrightarrow{a(x)} P \mid !\,G\sigma(x).P$. Then $!\,G\sigma(x).P \xrightarrow[[\,G:\mathcal{N}\,]]{G(x)} P_2$

by SREP, and $\mathbf{e}([\,G : \mathcal{N}\,]\sigma)$.

ALP Assume that $P\sigma \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P'$ and $(P \xrightarrow{\mu} P') =_\alpha (P \xrightarrow{(\nu\tilde{b}\rho)\,\overline{a}\,M\rho} P'\rho)$ where $\rho : \{\tilde{b}\} \to \mathcal{N}$ is injective such that $(\mathrm{range}(\rho) \setminus \{\tilde{b}\}) \cap (\mathrm{n}(a, M) \cup \mathrm{fn}(P)' \cup \mathrm{n}(\phi)) = \emptyset$.
By induction there are $\widetilde{c}, P_2, \phi, \mu_s$ such that $\mathrm{bn}(\mu_s) = \{\widetilde{c}\} \uplus \{\tilde{b}\}$, $\mu = \mathbf{e}(\mu_s\sigma)$, $[\![\phi\sigma]\!]$ and $P \xrightarrow[\phi]{\mu_s} P_2$ where $P_2\sigma >_\mathrm{a} P_1$ and $[\![\phi\sigma]\!]$.
By SALP we may choose $\{\tilde{c}\} \cap (\mathrm{range}(\rho)) = \emptyset$. Then $(P \xrightarrow[\phi]{\mu_s} P') =_\alpha (P \xrightarrow[\phi\rho]{\mu_s\rho} P'\rho)$, $P_2\rho\sigma >_\mathrm{a} P_1\rho$ and $[\![\phi\rho\sigma]\!]$.

□

Proposition A.2.3 ***(Proposition 4.1.24)***

We define $\sigma@\mu_s$ where μ_s is a symbolic late input action such that $\mathrm{bn}(\mu_s) \cap \mathrm{n}(\mathrm{range}(\sigma)) = \emptyset$ as the concrete early input action defined as follows: $\sigma@(\nu\tilde{b})\,\tau = \tau$, $\sigma@(\nu\tilde{b})\,\overline{F}\,G = (\nu\{b_i \mid b_i \in \mathrm{n}(\mathbf{e}(F\sigma))\})\,\overline{\mathbf{e}(F\sigma)}\;\mathbf{e}(G\sigma)$ and $\sigma@(\nu\tilde{b})\,F(x) = \mathbf{e}(F\sigma)\,\sigma(x)$. We then have that

1. *If $(pe, P) \xrightarrow{\tilde{\mu}} (pe', P')$ and $pe' \vdash \sigma$ then $P_i \xrightarrow{\sigma@\mu_i} P_{i+1}$ for $0 \le i \le n$ and $P'\sigma >_\mathrm{a} P_{n+1}$, where $P_0 = P\sigma$ and $\tilde{\mu} = \mu_0\mu_1 \cdots \mu_n$.*
2. *If $\mathrm{fn}(P) \subseteq \mathrm{dom}(tn)$, $pe \vdash \sigma$ and $P\sigma \xrightarrow{\tilde{\mu}} P_{n+1}$ such that $(\mathrm{dom}(tn) \cup \mathrm{fn}(P\sigma)) \cap \mathrm{bn}(\tilde{\mu}) = \emptyset$ and $\mathrm{bn}(\mu_i) \cap \mathrm{bn}(\mu_j) = \emptyset$ for $0 \le i < j \le n$ then $(pe, P) \xrightarrow{\tilde{\mu}'} (pe', P')$ such that $\exists\rho : pe' \vdash \rho$, $\forall v \in \mathrm{dom}(tv) : \rho(v) = \sigma(v)$, $P'\rho >_\mathrm{a} P_{n+1}$ and $\mu_i = \sigma@\mu'_i$ for $0 \le i \le n$ where $\tilde{\mu} = \mu_0\mu_1 \cdots \mu_n$ and $\tilde{\mu}' = \mu'_0\mu'_1 \cdots \mu'_n$.*

PROOF:

1. By induction on $|\widetilde{\mu}| = n$. If $n = 0$ there is nothing to prove. If $n = k + 1$ we have that $(pe, P) \xrightarrow{\tilde{\mu}} (pe', P') \xrightarrow{\mu} (pe'', P'')$ and $pe'' \vdash \sigma$ then $P' \xrightarrow[(\nu\tilde{c})\,\phi]{\mu}_s P''$ where
   - $\mathrm{bn}(\mu) \cap \mathrm{dom}(tn') = \emptyset$
   - $tn'' = tn' \cup \{x \mapsto i + 1 \mid x \in \mathrm{bn}(\mu)\}$
   - $tv'' = tv' \cup \{x \mapsto i + 1 \mid x \in \mathrm{bv}(\mu)\}$
   - $\psi'' = \psi' \wedge \phi$
   - $i = \max(\mathrm{range}(tn') \cup \mathrm{range}(tv') \cup \{0\})$

   This gives that $pe' \vdash \sigma$, so by induction $P_i \xrightarrow{\sigma@\mu_i} P_{i+1}$ for $0 \le i \le k$ and $P'\sigma >_\mathrm{a} P_{k+1}$, where $P_0 = P\sigma$ and $\tilde{\mu} = \mu_0\mu_1 \cdots \mu_k$. Since $\mathrm{n}(\mathrm{range}(\sigma)) \cap (\{\tilde{c}\} \cup \mathrm{bn}(\mu)) = \emptyset$ Theorem 4.1.18.1 gives that $P'\sigma \xrightarrow{\mathbf{e}(\mu\sigma)} Q$ and $P''\sigma >_\mathrm{a} Q$. By Lemma 4.1.7 we then have that $P_{k+1} \xrightarrow{\mathbf{e}(\mu\sigma)} Q'$ where $Q >_\mathrm{a} Q'$. By transitivity $P''\sigma >_\mathrm{a} Q'$. We now have three different cases for μ.

τ**:** Then $\mathbf{e}(\mu\sigma) = \tau = \sigma@\mu$.

$(\nu\tilde{b})\,\overline{F}\,G$**:** Then $\mathbf{e}(\mu\sigma) = (\nu\,\mathrm{en}(\mathbf{e}_\mathrm{a}(F)))\,\overline{\mathbf{e}(F\sigma)}\,\mathbf{e}(G\sigma) = \sigma@\mu$ since $\{b_i \mid b_i \in \mathrm{n}(\mathbf{e}(F\sigma))\} = \mathrm{en}(\mathbf{e}_\mathrm{a}(F))$ by Lemma 4.1.16.2.

$F(x)$**:** Then $\mathbf{e}(\mu\sigma) = \mathbf{e}(F\sigma)(x)$ and $\sigma@\mu = \mathbf{e}(F\sigma)\,\sigma(x)$. Moreover, $P_{k+1} \xrightarrow{\mathbf{e}(F\sigma)\,\sigma(x)} Q'\{{}^{\sigma}/_{x}\}$. Since σ is idempotent and $\mathrm{dom}(\sigma)\cap\mathrm{fn}(Q') \subseteq \{x\}$ we have that $P''\sigma >_\mathrm{a} Q'\{{}^{\sigma(x)}/_{x}\}$.

2. By induction on n. If $n = -1$ we take $\rho = \sigma$. If $n = k+1$ and $pe \vdash \sigma$ and $P\sigma \xrightarrow{\tilde{\mu}} P_{k+1} \xrightarrow{\mu_{k+1}} P_{k+2}$ such that $\mathrm{dom}(tn) \cap \mathrm{bn}(\tilde{\mu}\mu_{k+1}) = \emptyset$ and $\mathrm{bn}(\mu_i) \cap \mathrm{bn}(\mu_j) = \emptyset$ for $0 \leq i < j \leq k+1$ then by induction $(pe, P) \xrightarrow{\tilde{\mu}'} (pe', P')$ and ρ' satisfies $pe' \vdash \rho'$, $\forall v \in \mathrm{dom}(tv) : \rho'(v) = \sigma(v)$, $P'\rho' >_\mathrm{a} P_{k+1}$ and $\mu_i = \sigma@\mu'_i$ for $0 \leq i \leq n$ where $\tilde{\mu} = \mu_0\mu_1\cdots\mu_k$ and $\tilde{\mu}' = \mu'_0\mu'_1\cdots\mu'_k$. We have three different cases for μ_{k+1}.

   τ**:** Since $P'\rho' >_\mathrm{a} P_{k+1}$ we have by Lemma 4.1.7 that $P'\rho' \xrightarrow{\tau} P''$ such that $P'' >_\mathrm{a} P_{k+2}$. By Theorem 4.1.18.2 we have that $P' \xrightarrow[\phi]{(\nu\tilde{c})\,\tau}_s Q$ where $[\![\phi\sigma]\!]$ and $Q\sigma >_\mathrm{a} P''$. By Lalp-grd we may choose $\tilde{c} \cap \mathrm{dom}(tn) = \emptyset$. Then we have that $(pe', P') \xrightarrow{(\nu\tilde{c})\,\tau} (tn', tv', \psi' \wedge \phi, Q)$ where $Q\rho >_\mathrm{a} P_{k+2}$ by transitivity. For ρ we can then choose ρ'.

   $(\nu\tilde{b})\,\overline{a}\,M$**:** Since $P'\rho' >_\mathrm{a} P_{k+1}$ Lemma 4.1.7 gives that $P'\rho' \xrightarrow{(\nu\tilde{b})\,\overline{a}\,M} P''$ such that $P'' >_\mathrm{a} P_{k+2}$. By Theorem 4.1.18.2 we have $P' \xrightarrow[\phi]{(\nu\tilde{c})\,\overline{F}\,G}_s Q$ where $\mu_{k+1} = \mathbf{e}((\nu\tilde{c})\,\overline{F}\,G) = \sigma@(\nu\tilde{c})\,\overline{F}\,G$, $[\![\phi\sigma]\!]$ and $Q\sigma >_\mathrm{a} P''$. By Lalp-grd we may choose $\tilde{c} \cap \mathrm{dom}(tn) = \emptyset$. Then we have that $(pe', P') \xrightarrow{(\nu\tilde{c})\,\overline{F}\,G} (tn' \cup \{\tilde{b} \mapsto i+1\}, tv', \psi' \wedge \phi, Q)$ where $Q\rho >_\mathrm{a} P_{k+2}$ by transitivity. For ρ we can then choose ρ'.

   $a\,M$**:** Then $P_{k+1} \xrightarrow{a(x)} Q$ where $P_{k+2} = Q\{{}^{M}/_{x}\}$. Since $P'\rho' >_\mathrm{a} P_{k+1}$ we have by Lemma 4.1.7 that $P'\rho' \xrightarrow{a(x)} P''$ such that $P'' >_\mathrm{a} Q$. By Theorem 4.1.18.2 we have that $P' \xrightarrow[\phi]{(\nu\tilde{c})\,F(x)}_s Q'$ where $\mu_{k+1} = \mathbf{e}(F\sigma)(M)$, $[\![\phi\sigma]\!]$ and $Q'\sigma >_\mathrm{a} P''$. By Lalp-grd we may choose $\tilde{c} \cap (\mathrm{n}(M)\,\mathrm{dom}(tn)) = \emptyset$.

   Then $(pe', P') \xrightarrow{(\nu\tilde{c})\,F(x)} (tn', tv' \cup \{x \mapsto i+1\}, \psi' \wedge \phi, Q')$ where $i = max(\mathrm{range}(tv) \cup \mathrm{range}(tn))$.

   Now let $\sigma' : (\mathrm{n}(\psi' \wedge \phi) \setminus (\mathrm{dom}(tv') \cup \mathrm{dom}(tn'))) \to \mathcal{N}$ be injective such that $\mathrm{range}(\sigma') \cap (\mathrm{fn}(P\sigma) \cup \mathrm{n}(M) \cup \mathrm{n}(\mathrm{range}(\rho)) \cup \mathrm{dom}(tv') \cup \mathrm{dom}(tn') \cup \mathrm{n}(\phi)) = \emptyset$.

   By Lemma 4.1.8 $[\![(\psi' \wedge \phi)\sigma'\rho']\!]$, so $(tn, tv, \psi\sigma, P) \xrightarrow{\tilde{\mu}'F(x)} (tn', tv' \cup \{x \mapsto i+1\}, (\psi' \wedge \phi)\sigma', Q')$.

   For ρ we then choose $\rho'\{{}^{M}/_{x}\}$. We then have that $(tn', tv' \cup \{x \mapsto$

$i+1\}, (\psi' \wedge \phi)\sigma') \vdash \rho$, $\rho@F(x) = a\,M$ and $Q'\rho >_{\mathrm{a}} P_{k+2}$ by transitivity.

□

Lemma A.2.4 ***(Lemma 4.2.15)*** *Hedged bisimulation is sound up to bijective renaming and labelled bisimulation. In particular, if $\mathcal{R}$ is a hedged bisimulation up to bijective renaming and labelled bisimulation then $\mathcal{R}_{\mathrm{bl}}$ is a hedged bisimulation.*

Proof: We first note that "up to injective renaming" and "up to labelled bisimilarity" commute and are both idempotent. If σ is a bijective renaming $\mathcal{N} \to \mathcal{N}$ we write σ^{-1} for its inverse.

Let $\mathcal{R}$ be a hedged bisimulation up to bijective renaming and labelled bisimulation. Assume that $h \vdash P\ \mathcal{R}_{\mathrm{bl}}\ Q$ and that $P \xrightarrow{\mu} P'$. By the definition of $\mathcal{R}_{\mathrm{bl}}$, there are P_{bl}, Q_{bl} and bijective $\sigma : \mathcal{N} \to \mathcal{N}$ and $\rho : \mathcal{N} \to \mathcal{N}$ such that $P_{bl}\sigma \sim P$, $Q_{bl}\rho \sim Q$, and $h' \vdash P_{bl}\ \mathcal{R}\ Q_{bl}$ where $h' = \{(M\sigma^{-1}, N\rho^{-1}) \mid (M, N) \in h\}$.

1. Assume that $\mu = \tau$. By bisimilarity $P_{bl}\sigma \xrightarrow{\tau} P'_{bl} \sim P'$. Since σ is bijective $P_{bl} \xrightarrow{\tau} P'_{bl}\sigma^{-1} \sim P'\sigma^{-1}$.
   Since $\mathcal{R}$ is a hedged bisimulation up to bijective renaming and labelled bisimulation there is Q'_{bl} such that $Q_{bl} \xrightarrow{\tau} Q'_{bl}$ and $h' \vdash P'_{bl}\sigma^{-1}\ \mathcal{R}_{\mathrm{bl}}\ Q'_{bl}$.
   Since ρ is bijective $Q_{bl}\rho \xrightarrow{\tau} Q'_{bl}\rho$. By bisimilarity $Q \xrightarrow{\tau} Q'$ with $Q'_{bl}\rho \sim Q'$, so $h \vdash P'\ \mathcal{R}_{\mathrm{blbl}}\ Q'$ where $\mathcal{R}_{\mathrm{blbl}} = \mathcal{R}_{\mathrm{bl}}$ by commutativity and idempotence.
2. Assume that $\mu = a(x)$, and $h \vdash a \leftrightarrow b$ and $B \subset \mathcal{N}$ is finite such that $h \cup \mathrm{Id}_B \vdash M \leftrightarrow N$ and $B \cap (\mathrm{fn}(P, Q) \cup \mathrm{n}(h)) = \emptyset$.
   Since B are not necessarily fresh for P_{bl}, Q_{bl}, we need to invent another set of fresh names. Let $C \subset \mathcal{N}$ such that $C \cap (\mathrm{n}(h, \sigma, \rho) \cup \mathrm{fn}(P_{bl}, Q_{bl}, P, Q)) = \emptyset$ and $|C| = |B|$. Letting $\eta : B \to C$ be any bijective function, we get $h' \cup \mathrm{Id}_C \vdash M\sigma^{-1}\eta \leftrightarrow N\rho^{-1}\eta$.
   By bisimilarity $P_{bl}\sigma \xrightarrow{a(x)} P'_{bl}$ with $P'_{bl}\{^{M'}/_x\} \sim P'\{^{M'}/_x\}$ for all $M' \in \mathcal{M}$. Since σ is bijective $P_{bl} \xrightarrow{a\sigma^{-1}(x)} P'_{bl}\sigma^{-1}$ with $P'_{bl}\sigma^{-1}\{^{M\sigma^{-1}\eta}/_x\} \sim P'\sigma^{-1}\{^{M\sigma^{-1}\eta}/_x\}$.
   Since $\mathcal{R}$ is a hedged bisimulation up to bijective renaming and labelled bisimulation there is Q'_{bl} such that $Q_{bl} \xrightarrow{b\rho-1(x)} Q'_{bl}$ and $h' \cup \mathrm{Id}_C \vdash P'_{bl}\sigma^{-1}\{^{M\sigma^{-1}\eta}/_x\}\ \mathcal{R}_{\mathrm{bl}}\ Q'_{bl}\{^{N\sigma^{-1}\eta}/_x\}$. By applying η^{-1} we get $h' \cup \mathrm{Id}_B \vdash P'_{bl}\{^{M}/_x\}\sigma^{-1}\ \mathcal{R}_{\mathrm{blb}}\ Q'_{bl}\{^{N\rho^{-1}}/_x\}$.
   Since ρ is bijective $Q_{bl}\rho \xrightarrow{b(x)} Q'_{bl}\rho$. By bisimilarity $Q \xrightarrow{b(x)} Q'$ with $Q'_{bl}\rho\{^{N}/_x\} \sim Q'\{^{N}/_x\}$. Thus $h \cup \mathrm{Id}_B \vdash P'\{^{M}/_x\}\ \mathcal{R}_{\mathrm{blbbl}}\ Q'\{^{N}/_x\}$ where $\mathcal{R}_{\mathrm{blbbl}} = \mathcal{R}_{\mathrm{bl}}$ by commutativity and idempotence.
3. Assume that $\mu = (\nu\tilde{c})\,\overline{a}\,M$, $h \vdash a \leftrightarrow b$ and $\{\tilde{c}\} \cap (\mathrm{fn}(P) \cup \mathrm{n}(\pi_1(h))) = \emptyset$.

As above, we may not have $\{\tilde{c}\}$ fresh for P_{bl}. We take $\widetilde{c'}$ with $\{\tilde{c'}\} \cap (\mathrm{fn}(P, P_{bl}) \cup \mathrm{n}(\sigma, \pi_1(h))) = \emptyset$, $|\widetilde{c}| = |\widetilde{c'}|$ and a bijective $\eta : \{\tilde{c}\} \to \{\tilde{c'}\}$. By ALP
$P \xrightarrow{(\nu \tilde{c'})\,\overline{a}\,M\eta} P'\eta$.

By bisimilarity $P_{bl}\sigma \xrightarrow{(\nu \tilde{c'})\,\overline{a}\,M\eta} P'_{bl} \sim P'\eta$. Since σ is bijective
$P_{bl} \xrightarrow{(\nu \tilde{c'})\,\overline{a\sigma^{-1}}\,M\eta\sigma^{-1}} P'_{bl}\sigma^{-1} \sim P'\eta\sigma^{-1}$.
Since $\mathcal{R}$ is a hedged bisimulation up to bijective renaming and labelled bisimulation there are $Q'_{bl}, \widetilde{d}, N$ such that $\{\tilde{d}\} \cap (\mathrm{fn}(Q) \cup \mathrm{n}(\pi_2(h))) = \emptyset$,
$Q_{bl} \xrightarrow{(\nu \tilde{d})\,\overline{b\rho^{-1}}\,N} Q'_{bl}$ and $h' \vdash P'_{bl}\sigma^{-1}\ \mathcal{R}_{\mathrm{bl}}\ Q'_{bl}$.
Again, we may not have $\{\tilde{d}\}$ fresh for Q. We take $\widetilde{d'}$ with $\{\tilde{d'}\} \cap (\mathrm{fn}(Q, Q_{bl}) \cup \mathrm{n}(\rho, \pi_2(h))) = \emptyset$, $|\widetilde{d}| = |\widetilde{d'}|$ and a bijective $\zeta : \{\tilde{d}\} \to \{\tilde{d'}\}$. By ALP
$Q_{bl} \xrightarrow{(\nu \tilde{d'})\,\overline{b\rho_{-1}}\,N\zeta} Q'_{bl}\zeta$.

Since ρ is bijective $Q_{bl}\rho \xrightarrow{(\nu \tilde{d'})\,\overline{b\rho_{-1}}\,N\zeta} Q'_{bl}\zeta\rho$. By bisimilarity $Q \xrightarrow{(\nu \tilde{d'})\,\overline{b\rho_{-1}}\,N\zeta} Q'$ with $Q'_{bl}\zeta\rho \sim Q'$, so $h \vdash P'\ \mathcal{R}_{\mathrm{blbl}}\ Q'$ where $\mathcal{R}_{\mathrm{blbl}} = \mathcal{R}_{\mathrm{bl}}$ by commutativity and idempotence.

Then $\mathcal{R} \subseteq \sim_{\mathrm{h}}$ since $\mathcal{R} \subseteq \mathcal{R}_{\mathrm{bl}}$ because b and l are both expansions. □

Appendix B

A Prototype Implementation

As it stands, the definition of symbolic bisimulation is not directly mechanizable. The definition of environment consistency, in particular, contains several universal quantifications over environment-respecting substitution pairs. During the course of our work, we implemented a prototype that checks for a slightly more concrete notion of bisimulation (some variable instantiations are made) for a simple expression language guard language without negation.

Language and Semantics Inspired by the *Mobility Workbench* [VM94], we add abstractions and concretions to the language. Processes are kept in a standard form and de Bruijn indices [Hir99] are used to handle bound names. To simplify the writing of processes we encoded a **let** construct using abstractions and $[\cdot : \mathcal{M}]$.

Semantics There are no real difficulties to adapt the symbolic semantics to abstractions and concretions. The most tricky part is to manage de Bruijn indices properly.

Environments The usual functions (synthesis, analysis et c.) on hedges are easy to implement. It is also straightforward to check that a pair of substitutions (σ, ρ) respects a symbolic environment se. In this prototype, we did not implement environment decompositions, yielding some incompletess wit hrespect to barbed equivalence.

The major problem for mechanization is verifying environment consistency. To check if all concretizations of an environment are consistent, it is not sufficient to consider only the most general unifiers, due to the presence of complex keys (c.f. Example 4.3.3). We used a brute-force approach, calculating decryption conditions for all message pairs in the environment, intended to reduce this problem to formula equivalence. However, we did not acheive a correctness proof for our algorithm (a recent proof of a similar result takes up 90 pages in [Bau07]).

Bisimulation For the bisimulation, we need to check if a transition is detectable and possible. Using the same algorithm as for checking the consistency of all concretizations of an environment, we calculated conditions under which there is a concretization that knows the channel. We also have that a transition is impossible iff its transition guard is equivalent to false.

We successfully validated our prototype on the subset of protocols of the Security Protocols Online REpository [Cac] that only use symmetric encryption. Our prototype was used in conjunction with the then-current version of Spyer [BN07] and an ad-hoc tool for generating specifications in process format from implementations, inspired by the examples of [AG98]. Symbolic bisimilarity improves considerably on the size of the bisimulation relations compared to the decision procedure of [Hüt02] (which also can be used for concrete hedged bisimilarity). Using our prototype implementation, there are 219 states in the bisimulation showing the secrecy of the Wide-mouthed Frog protocol of [AG99] with one participant. In contrast, the decision procedure mentioned above generates $\gg 2^{2^{20}}$ branches for *each* of the five inputs of a run of the protocol.

Bibliography

[AB05] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1):102–146, 2005.

[AC04a] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Proceedings of ICALP '04*, volume 3142 of *LNCS*, pages 46–58. Springer, 2004.

[AC04b] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. Technical Report RR-5169, INRIA, 2004.

[AC06] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1-2):2–32, 2006.

[ACD07] Mathilde Arnaud, Véronique Cortier, and Stéphanie Delaune. Combining algorithms for deciding knowledge in security protocols. In Boris Konev and Frank Wolter, editors, *Proceedings of FroCoS '07*, volume 4720 of *LNCS*, pages 103–117. Springer, 2007.

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of POPL '01*, pages 104–115, 2001.

[AG98] Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The Spi calculus. *Journal of Information and Computation*, 148(1):1–70, 1999.

[AH85] Charles Antony and Tony Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[AL00] Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In Catuscia Palamidessi, editor, *Proceedings of CONCUR '00*, pages 380–394, 2000.

[AN95] Ross Anderson and Roger Needham. Programming Satan's computer. In Jan Wan Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*, pages 426–441. Springer, 1995.

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[Bau07] Mathieu Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. PhD thesis, École Normale Supérieure de Cachan, 2007.

[BB05] Michele Boreale and Maria Grazia Buscemi. A method for symbolic analysis of security protocols. *Theoretical Computer Science*, 338(1-3):393–425, 2005.

[BBN04] Johannes Borgström, Sébastien Briais, and Uwe Nestmann. Symbolic bisimulation in the spi calculus. In Philippa Gardner and Nobuko Yoshida, editors, *Proceedings of CONCUR '04*, volume 3170 of *LNCS*, pages 161–176. Springer, 2004.

[BD96] Michele Boreale and Rocco De Nicola. A symbolic semantics for the π-calculus. *Information and Computation*, 126(1):34–52, 1996.

[BDP99] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. In Giuseppe Longo, editor, *Proceedings of LICS '99*, pages 157–166. IEEE, Computer Society Press, July 1999.

[BDP02] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. *SIAM Journal on Computing*, 31(3):947–986, 2002.

[BFGP03] Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Riccardo Pucella. Tulafale: A security tool for web services. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *Proceedings of FMCO '03*, volume 3188 of *LNCS*, pages 197–222. Springer, 2003.

[BFGT06] Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Stephen Tse. Verified interoperable implementations of security protocols. In *Proceedings of CSFW '06*, pages 139–152. IEEE Computer Society, 2006.

[BG02] Michele Boreale and Daniele Gorla. On compositional reasoning in the spi-calculus. In Mogens Nielsen and Uffe Engberg, editors, *Proceedings of FoSSaCS '02*, volume 2303 of *LNCS*, pages 67–81. Springer, 2002.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proceedings of CSFW'01*, pages 82–96. IEEE, 2001.

[BN07] Sébastien Briais and Uwe Nestmann. A formal semantics for protocol narrations. *Theoretical Computer Science*, 389(3):484–511, 2007.

[Bor95] Michele Boreale. *Process Algebraic Theories for Mobile Systems*. PhD thesis, Università degli Studi di Roma "La Sapienza", 1995.

[Bor01] Michele Boreale. Symbolic trace analysis of cryptographic protocols. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Proceedings of ICALP '01*, pages 667–681, 2001.

[Bor04] Micele Boreale. Erratum of Proof techniques for cryptographic processes. Unpublished manuscript, August 2004.

[BPV05] Michael Baldamus, Joachim Parrow, and Björn Victor. A fully abstract encoding of the pi-calculus with data terms. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proceedings of ICALP '05*, volume 3580 of *LNCS*, pages 1202–1213. Springer, 2005.

[Bri08] Sébastien Briais. *Theory and Tool Support for the Formal Verification of Cryptographic Protocols*. PhD thesis, EPFL, 2008.

[Cac] LSV Cachan. Security Protocols Online REpository. Accessed on February 23, 2008, `http://www.lsv.ens-cachan.fr/spore/`.

[CDL06] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[CKRT03] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In Paritosh K. Pandya and Jaikumar

Radhakrishnan, editors, *Proceedings of FSTTCS '03*, volume 2914 of *LNCS*, pages 124–135. Springer, 2003.

[Cor02] Veronique Cortier. Observational equivalence and trace equivalence in an extension of spi-calculus. Technical Report LSV-02-3, Laboratoire de Specification and Verification, ENS de Cachan, March 2002.

[Cor03] Véronique Cortier. *Vérification automatique des protocoles cryptographiques*. PhD thesis, École Normale Supérieure de Cachan, 2003.

[CRZ07] Véronique Cortier, Michaël Rusinowitch, and Eugen Zalinescu. Relating two standard notions of secrecy. *Logical Methods in Computer Science*, 3, 2007.

[CS02] Hubert Comon and Vitali Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not? *Journal of Telecommunications and Information Technology*, 4:5–15, 2002.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of CSFW'06*, pages 28–39. IEEE Computer Society Press, July 2006.

[DKR07] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Symbolic bisimulation for the applied pi-calculus. In V. Arvind and Sanjiva Prasad, editors, *Proceedings of FSTTCS'07*, LNCS. Springer, December 2007. To appear.

[dNH84] Rocco de Nicola and Matthew Hennessy. Testing equivalence for processes. *Theoretical Computer Science*, 34:81–133, 1984.

[DSV03] Luca Durante, Riccardo Sisto, and Adriano Valenzano. Automatic testing equivalence verification of spi-calculus specifications. *ACM Transactions on Software Engineering and Methodology*, 12(2):222–284, April 2003.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[EHHO99] Anders Strandløv Elkjær, Micheal Höhle, Hans Hüttel, and Kasper Overgaard. Towards automatic bisimilarity checking in the spi calculus. In *Combinatorics, Computation & Logic*, volume 21(3) of *Australian Computer Science Communications*, pages 175–189. Springer, January 1999.

[FA01] Marcelo Fiore and Martín Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proceedings of CSFW '01*, pages 160–173, 2001.

[FHJ01] Ulrik Frendrup, Hans Hüttel, and Jesper Nyholm Jensen. Two notions of environment sensitive bisimilarity for spi-calculus processes. Unpublished manuscript, 2001.

[GJ04] Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3-4):435–483, 2004.

[GJ05] Andrew D. Gordon and Alan Jeffrey. Secrecy despite compromise: Types, cryptography, and the pi-calculus. In Martín Abadi and Luca de Alfaro, editors, *Proceedings of CONCUR '05*, volume 3653 of *LNCS*, pages 186–201. Springer, 2005.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[Hir99] Daniel Hirschkoff. *Mise en Œuvre de Preuves de Bisimulation*. PhD thesis, École Nationale des Ponts et Chaussées, 1999.

[HJ06] Christian Haack and Alan Jeffrey. Pattern-matching spi-calculus. *Information and Computation*, 204(8):1195–1263, 2006.

[HL95] Matthew Hennessy and Huimin Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[Hui99] Antti Huima. Efficient infinite-state analysis of security protocols. In *FLOC Workshop on Formal Methods and Security Protocols*, 1999.

[Hüt02] Hans Hüttel. Deciding framed bisimilarity. In Antonín Kučera and Richard Mayr, editors, *Pre-proceedings of INFINITY '02*, pages 1–20, june 2002.

[JV07] Magnus Johansson and Björn Victor. A fully abstract symbolic semantics for the applied pi-calculus. Unpublished manuscript, May 2007.

[KM07] Temesghen Kahsai and Marino Miculan. Spi calculus in Isabelle/HOL-Nominal. http://www.cs.swan.ac.uk/~csteme/SpiInIsabelle/SpiInIsabelle.html, 2007.

[KMM94] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[KR04] Steve Kremer and Mark D. Ryan. Analysing the vulnerability of protocols to produce known-pair and chosen-text attacks. *ENTCS*, 128(5):87–104, 2004. Proceedings of SecCo '04.

[LCFW05] Yinhua Lü, Xiaorong Chen, Luming Fang, and Hangjun Wang. Towards a symbolic bisimulation for the spi calculus. In Xiaohua Jia, Jie Wu, and Yanxiang He, editors, *Mobile Ad-Hoc and Sensor Networks*, volume 3794 of *LNCS*, pages 1095–1102. Springer, 2005.

[Lin94] Huimin Lin. Symbolic bisimulations and proof systems for the pi-calculus. Technical Report 94:07, COGS, University of Sussex, 1994.

[Liu94] Xinxin Liu. Characterizing bisimulation congruence in the pi-calculus (extended abstract). In Bengt Jonsson and Joachim Parrow, editors, *Proceedings of CONCUR '94*, volume 836 of *LNCS*, pages 331–350. Springer, 1994.

[LMBG05] Kevin D. Lux, Michael J. May, Nayan L. Bhattad, and Carl A. Gunter. WSEmail: Secure internet messaging based on web services. In *Proceedings of ICWS '05*, pages 75–82. IEEE Computer Society, 2005.

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Tiziana Margaria and Bernhard Steffen, editors, *Proceedings of TACAS '96*, volume 1055 of *LNCS*, pages 147–166. Springer, 1996.

[Mar99] Fabio Martinelli. *Formal methods for the analysis of open systems with applications to security properties*. PhD thesis, University of Siena, 1999.

[Mar02] Fabio Martinelli. Symbolic semantics and analysis for crypto-ccs with (almost) generic inference systems. In Krzysztof Diks and Wojciech Rytter, editors, *Proceedings of MFCS '02*, volume 2420 of *LNCS*, pages 519–531. Springer, 2002.

[Mil80] Robin Milner. *A Calculus of Communicating Systems*. Springer, 1980.

[Mil81] Robin Milner. A modal characterisation of observable machine-behaviour. In Egidio Astesiano and Corrado Böhm, editors, *Proceedings of CAAP '81*, volume 112 of *LNCS*, pages 25–34. Springer, 1981.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Mil99] Robin Milner. *Communicating and Mobile Systems: The pi-Calculus*. Cambridge University Press, 1999.

[Mit01] John C. Mitchell. Probabilistic polynomial-time process calculus and security protocol analysis. In David Sands, editor, *Proceedings of ESOP 2001*, volume 2028 of *LNCS*, pages 23–29. Springer, 2001.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Journal of Information and Computation*, 100:1–77, September 1992.

[MPW93] Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114:149–171, 1993.

[MS92] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *Proceedings of ICALP '92*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.

[Par81] David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.

[Pau98] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[Ros97] A. William Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1997.

[San96] Davide Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996.

[San98] Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, 1998.

[Sch96] Steve Schneider. Security properties and CSP. In *Proceedings of SP '96*, pages 174–187, Washington, DC, USA, 1996. IEEE Computer Society.

[SR01] Steve A. Schneider and Peter Ryan. *The Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

[SW01] Davide Sangiorgi and David Walker. *The pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[Tiu07] Alwen Tiu. A trace based bisimulation for the spi calculus: An extended abstract. In Zhong Shao, editor, *Proceedings of APLAS '07*, volume 4807 of *LNCS*, pages 367–382. Springer, 2007.

[VM94] Björn Victor and Faron Moller. The Mobility Workbench — a tool for the π-calculus. In David L. Dill, editor, *Proceedings of CAV '94*, volume 818 of *LNCS*, pages 428–440. Springer, 1994.

[WL93] Tomac Y.C. Woo and Simon S. Lam. A semantic model for authentication protocols. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pages 178–194, 1993.

Curriculum Vitæ

Personal Information

Full name: Eric Johannes Borgström
Birth date: 1978-12-29
Swedish citizen, born in Sweden.

Education

I hold a Master of Science in Engineering from the Royal Institute of Technology (KTH), Stockholm, awarded in June 2003. My field of studies was *Computer Science and Computer Engineering*.

School	Subject	Period	ECTS credits
KTH, Stockholm	Computer science	1998-09 – 2003-06	186
EPFL, Lausanne	Computer science	2000-10 – 2001-07	58
Stockholm University	Mathematics	1998-04 – 2000-02	97.5
Stockholm University	History of ideas	1998-09 – 1999-05	30

Full-time Professional Experience

Employer	Job description	Period
TU Berlin	Research Assistant	2005-10 –
Microsoft Research, Cambridge	Research Intern	2006-07 – 2006-09
EPFL, Lausanne	Research Assistant	2002-09 – 2005-09
Swedish military	R&D	1999-06 – 1999-09, 1998-06 – 1998-07
Military service	Communications	1997-06 – 1998-05

Publications

Johannes Borgström, Andrew D. Gordon and Andrew Phillips, *A Chart Semantics for the pi Calculus*. In *Proceedings of EXPRESS 2007*, ENTCS **194**(2), pages 3-29, Elsevier 2008.

Johannes Borgström, Simon Kramer and Olga Grinchtein, *Timed Calculus of Cryptographic Communication*. In *Proceedings of FAST 2006*, LNCS 4691, Springer 2007.

Johannes Borgström, Simon Kramer and Uwe Nestmann, *Calculus of Cryptographic Communication*. Presented at FCS-ARSPA 2006.

Johannes Borgström, *Static Equivalence* is *Harder than Knowledge*. In *Proceedings of EXPRESS 2005*, ENTCS **154**(3), pages 45-57, Elsevier 2006.

Johannes Borgström and Uwe Nestmann, *On Bisimulations for the Spi-calculus*. In *Mathematical Structures in Computer Science* **15**, pages 482-552. Cambridge University Press 2005. Short version in *Proceedings of AMAST 2002*, LNCS 2422, Springer 2002.

Johannes Borgström, Sébastien Briais and Uwe Nestmann, *Symbolic Bisimulation in the Spi Calculus.* In *Proceedings of CONCUR 2004*, LNCS 3170, Springer 2004.

Johannes Borgström, Uwe Nestmann, Luc Onana Alima and Dilian Gurov, *Verifying a Structured Peer-to-peer Overlay Network: The Static Case.* In *Proceedings of Global Computing 2004*, LNCS 3267, Springer 2004. Long version available as EPFL Technical report IC/2004/76.